دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

پایان‌نامه‌ی کارشناسی ارشد
گرایش مهندسی نرم‌افزار

عنوان:

تقریب مسئله‌ی kـ مرکز با نقاط پرت در مدل پنجره‌ی لغزان

نگارش:

علی مصطفوی

استاد راهنما:

دکتر حمید ضرابی زاده

مرداد ۱۳۹۷

بسم الله الرحمن الرحیم

به نام خدا

دانشگاه صنعتی شریف

دانشکده‌ی مهندسی کامپیوتر

پایان‌نامه‌ی کارشناسی ارشد

عنوان: تقریب مسئله‌ی kـ مرکز با نقاط پرت در مدل پنجره‌ی لغزان

نگارش: علی مصطفوی

کمیته‌ی ممتحنین

استاد راهنما: دکتر حمید ضرابی زاده امضاء:

استاد مشاور: دکتر محمد قدسی امضاء:

استاد مدعو: دکتر علی محدث خراسانی امضاء:

تاریخ:

چکیده

با افزایش چشم‌گیر حجم داده‌ها، امروزه برای بسیاری از مسئله‌ها ذخیره‌ی تمام داده‌های مورد پردازش در حافظه امکان‌پذیر نیست. این موضوع باعث ارائه‌ی مدل‌های جدید پردازش داده، از جمله مدل جریان داده شده است. در مدل جریان داده، فرض می‌کنیم که داده‌های مسئله به مرور زمان وارد شده و حافظه‌ی در دسترس بسیار کم‌تر از اندازه‌ی ورودی است. در برخی مسائل، داده‌های اخیر برای ما ارزش بیش‌تری دارند. برای این گونه مسائل، مدل پنجره‌ی لغزان ارائه شده است. در این مدل ما می‌خواهیم که یک جواب تقریبی برای n داده‌ی اخیر با استفاده از حافظه‌ای بسیار کم‌تر از n به دست آوریم. مسئله‌ی kـ مرکز یکی از مسائل مهم بهینه‌سازی است که هدف از آن یافتن k مرکز از رأس‌های یک گراف است طوری که بیش‌ترین فاصله‌ی رأس‌ها تا مرکزها کمینه شود. در دنیای واقعی در بسیاری از مسائل از جمله kـ مرکز، داده‌های پرت داریم که می‌توانند جواب ما را به شدت تحت تاثیر قرار دهند. برای مقابله با این مشکل، می‌توان بخشی از داده ها را به عنوان داده‌های پرت در پیدا کردن جواب بهینه لحاظ نکرد. در این پژوهش، قصد داریم مسئله‌ی kـ مرکز با وجود داده‌های پرت را در مدل پنجره‌ی لغزان بررسی کنیم و برای این مسئله یک الگوریتم تقریبی با ضریب تقریب ثابت ارائه کنیم.

ابتدا مسئله‌ی ۱ـ مرکز با نقاط پرت در مدل پنجره‌ی لغزان را در فضاهای اقلیدسی و متریک بررسی می‌کنیم و برای آن‌ها به ترتیب الگوریتم‌هایی با ضریب تقریب ۲ و $\epsilon +$ ۳ ارائه می‌دهیم. همچنین نشان می‌دهیم که چگونه می‌توان ضریب تقریب را در فضاهای متریک که بعد مضاعف آن‌ها ثابت است، به $\epsilon +$ ۲ کاهش داد. سپس یک الگوریتم تقریبی با ضریب تقریب $\epsilon +$ ۸ برای مسئله‌ی kـ مرکز با نقاط پرت در مدل پنجره‌ی لغزان ارائه می‌دهیم و نشان می‌دهیم که الگوریتم ما از نظر حافظه تقریباً بهینه است همچنین در حالتی که بعد مضاعف ثابت است، ضریب تقریب را به $\epsilon +$ ۳ کاهش می‌دهیم. در قسمت بعد نشان می‌دهیم که اگر علاوه بر شعاع بهینه، بر روی تعداد نقاط پرت هم تقریب بزنیم، می‌توان الگوریتمی دوتقریبی ارائه داد که حداکثر $\delta +$ ۱ برابر بهینه نقاط پرت دارد و شعاع آن حداکثر $\epsilon +$ ۱۴ برابر شعاع بهینه است. حافظه‌ی مورد نیاز این الگوریتم، از کران پایین اثبات شده برای حالت قبل کم‌تر است.

کلیدواژه‌ها: الگوریتم‌های تقریبی، فضای متریک، هندسه‌ی محاسباتی، kـ مرکز، پنجره‌ی لغزان، نقاط پرت

فهرست مطالب

فهرست شکل‌ها

فصل ۱

مقدمه

۱ـ۱ تعریف مسئله

در مسئله‌ی خوشه بندی[۱] ، می‌خواهیم داده‌های ورودی را به چند دسته تقسیم کنیم، به طوری که داده‌های هر دسته به هم شبیه باشند. یکی از روش‌های خوشه بندی به این صورت است که به هر دسته یک مرکز[۲] اختصاص دهیم و معیار شباهت هر دسته را شعاعی در نظر بگیریم که با آن، مرکز انتخاب شده می‌تواند همه‌ی نقاط آن دسته را پوشش دهد. هدف ما در مسئله‌ی kـ مرکز، پیدا کردن مرکزهایی است که بیشترین معیار شباهت را دارند. به عبارت دیگر، می‌خواهیم k نقطه را پیدا کنیم که با کم‌ترین شعاع همه‌ی نقاط ورودی را پوشش می‌دهند (یک معیار شباهت دیگر، جمع توان دوم فاصله‌ی نقاط تا نزدیک‌ترین مرکز به آن‌ها است. این مسئله kـ میانگین[۳] نام دارد).

یکی از مشکلات مسئله‌ی kـ مرکز، حساسیت شدید آن به نقاط پرت[۴] است. برای مقابله با این مشکل، مسئله‌ی kـ مرکز با نقاط پرت مطرح شد. در این نسخه از مسئله، ما اجازه داریم تعدادی از نقاط را به عنوان نقطه‌ی پرت در نظر بگیریم و آن‌ها را پوشش ندهیم.

یکی از مدل‌هایی که اخیراً مسئله‌ی kـ مرکز در آن بررسی شده است، مدل پنجره‌ی لغزان است که در آن، هدف این است که جواب را برای n داده‌ی اخیر با استفاده از حافظه‌ی زیرخطی در n محاسبه

[۱] clustering
[۲] center
[۳] mean
[۴] outliers

کنیم.

در این پایان نامه، مسئله‌ی kـ مرکز با نقاط پرت را در مدل پنجره‌ی لغزان[۵] بررسی می‌کنیم که در آن، هدف این است که جواب را برای n داده‌ی اخیر، با استفاده از حافظه‌ی زیر خطی (ترجیحاً لگاریتمی) در n محاسبه کنیم. در ادامه‌ی این فصل، مفاهیم اولیه و تعریف دقیق ریاضی مسئله را مطرح خواهیم کرد. در فصل دوم، خلاصه‌ای از کارهای پیشین انجام شده در این زمینه را شرح خواهیم داد و فصل سوم به نتایج جدید ما اختصاص دارد. در نهایت در فصل چهارم، جمع بندی و کارهای آتی را شرح خواهیم داد.

۱ـ۲ اهمیت موضوع

خوشه بندی، یکی از روش‌های اصلی مورد استفاده در یادگیری بدون نظارت[۶] است که در بسیاری از زمینه‌ها مثل تشخیص الگو[۷] ، تحلیل تصویر[۸] و فشرده سازی[۹] کاربرد دارد [۱، ۲، ۳]. متداول‌ترین روش مورد استفاده در عمل برای خوشه بندی، kـ میانگین است. یکی از دلایل استفاده از این روش به جای kـ مرکز، وجود یک الگوریتم ساده و کاربردی برای محاسبه‌ی kـ میانگین است [۴]. یک دلیل دیگر برای این موضوع، همان طور که اشاره شد، حساسیت شدید مسئله‌ی kـ مرکز به نقاط پرت است. ارائه‌ی الگوریتمی کارا برای مسئله‌ی kـ مرکز که نقاط پرت را هم در نظر بگیرد، می‌تواند استفاده از این روش را در کاربردهای یاد شده افزایش دهد.

۱ـ۳ ادبیات موضوع

اوّلین الگوریتم تقریبی برای مسئله‌ی kـ مرکز در فضای متریک، توسط گنزالز ارائه شد [۵] که ضریب تقریب ۲ داشت. در این مقاله نشان داده شد که ارائه دادن هر ضریب تقریب بهتر از ۲ برای این مسئله، NPـ سخت است. هاچبام و شمویس [۶] نیز با استفاده از روشی دیگر به ضریب تقریب ۲ دست یافتند.

[۵] sliding window
[۶] unsupervised learning
[۷] pattern recognition
[۸] image analysis
[۹] compression

در حالتی که نقاط پرت داریم، چاریکار و دیگران [۷] نشان دادند که با فرض $P \neq NP$، نمی‌توان مسئله را با ضریب تقریب بهتر از ۳ حل کرد. آن‌ها همچنین الگوریتمی با همین ضریب تقریب ارائه دادند.

مککاتچن و خولر [۸]، مسئله‌ی kـ مرکز را در مدل جویبار داده بررسی کردند و الگوریتمی با ضریب تقریب $\epsilon + ۲$ برای حالتی که نقاط پرت نداریم، و الگوریتمی با ضریب تقریب $\epsilon + ۴$ برای حالتی که نقاط پرت داریم ارائه دادند.

مسئله‌ی kـ مرکز برای اولین بار توسط کوهن اداد و دیگران [۹]، در مدل پنجره‌ی لغزان بررسی شد. آن‌ها الگوریتمی با ضریب تقریب $\epsilon + ۶$ برای این مسئله ارائه دادند و نشان دادند که هر الگوریتمی که تنها می‌تواند نقاط ورودی را ذخیره کند (یعنی قادر به تولید نقطه‌ی جدید نیست)، برای ارائه‌ی ضریب تقریب بهتر از ۴، باید حداقل $\Omega(\sqrt[3]{n})$ نقطه را ذخیره کند.

۱ـ۴ اهداف تحقیق

در این پایان نامه، مسئله‌ی kـ مرکز با نقاط پرت را در مدل پنجره‌ی لغزان بررسی می‌کنیم. این، اوّلین پژوهش برروی این مسئله در مدل پنجره‌ی لغزان است. تمرکز اصلی ما بر روی فضاهای متریک است امّا در بعضی مواقع، مسئله را در فضای اقلیدسی و یا فضای متریک با بعد مضاعف ثابت نیز بررسی می‌کنیم. همچنین سعی می‌کنیم کران پایینی برای حافظه‌ی مورد نیاز برای حل این مسئله ارائه دهیم.

۱ـ۵ ساختار پایان‌نامه

این پایان‌نامه شامل پنج فصل است. در فصل دوم، تعاریف و مفاهیم اوّلیه را شرح خواهیم داد. فصل سوم به بررسی کارهای پیشین اختصاص دارد. در فصل سوم، نتایج جدید را شرح خواهیم داد و در نهایت فصل پنجم به جمع بندی و کارهای آینده اختصاص دارد.

فصل ۲

مفاهیم اولیه

تعریف ۱ (فضای متریک[۱]). *به یک دوتایی (M,d) که $d : M \times M \to \mathbb{R}$ و شرایط زیر را داشته باشد فضای متریک می‌گویند.*

۱. $x = y \iff d(x,y) = ۰$

۲. $d(x,y) = d(y,x)$

۳. $d(x,z) \leqslant d(x,y) + d(y,z)$

در ادامه‌ی این پایان نامه از خاصیت سوم با نام نامساوی مثلثی[۲] یاد می‌کنیم. شاید آشنا ترین فضای متریک برای ما، فضای اقلیدسی[۳] سه بعدی باشد. به طور کلی فضای اقلیدسی d بعدی به این صورت تعریف می‌شود:

تعریف ۲ (فضای اقلیدسی d بعدی). *در این فضا $M = \mathbb{R}^d$ و*

$$d(x,y) = \|x-y\|_۲ = \sqrt{(x-y).(x-y)} = \sqrt{\sum_{i=۱}^{d} {(x-y)_i}^۲}$$

[۱] metric space
[۲] triangle inequality
[۳] euclidean space

در تعریف بالا، به $\|x\|_2$، L_2ــ نرم[۴] می‌گویند. به طور کلی L_pــ نرم به این صورت تعریف می‌شود:

$$\|x\|_p = \sqrt[p]{\sum_{1}^{d} |x_i|^p}$$

می‌توان نشان داد که فضای اقلیدسی، شرط‌های فضای متریک را ارضا می‌کند و در نتیجه یک فضای متریک است. از مهم ترین نرم‌ها، L_1 و L_∞ هستند. می‌توان نشان داد:

$$\|x\|_\infty = \max_i x_i$$

تعریف ۳ (توپ باز[۵]). *برای یک فضای متریک (M, d) مجموعه‌ی $\{m \in M | d(c, m) < r\}$ یک توپ باز به مرکز c و شعاع r گفته می‌شود.*

تعریف ۴ (توپ بسته[۶]). *برای یک فضای متریک (M, d) مجموعه‌ی $\{m \in M | d(c, m) \leqslant r\}$ یک توپ بسته به مرکز c و شعاع r گفته می‌شود.*

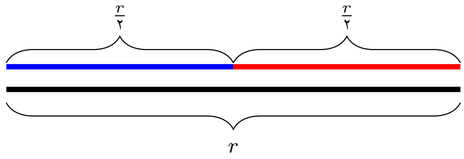
در ادامه‌ی این پایان نامه توپ بدون پسوند به معنی توپ بسته است.

تعریف ۵ (فضای مضاعف[۷]). *یک فضای متریک (M, d) فضای مضاعف گفته می‌شود اگر یک عدد ثابت $D > 0$ وجود داشته باشد طوری که هر توپ باز با شعاع r را بتوان با حداکثر D توپ باز با شعاع $\frac{r}{2}$ پوشش داد [۱۰]. به D بعد مضاعف[۸] این فضا می‌گوییم.*

تعریف ۶ (αــ نقطه‌ی میانی[۹]). *نقطه‌ی q برای مجموعه نقاط P در فضای اقلیدسی، αــ نقطه‌ی میانی گفته می‌شود اگر هر نیم صفحه‌ی گذرنده از q، شامل حداقل $\alpha|P|$ نقطه باشد.*

می‌توان نشان داد که هر مجموعه نقطه در فضای اقلیدسی d بعدی، یک $\frac{1}{d+1}$ نقطه‌ی میانی دارد که به آن نقطه‌ی میانی می‌گوییم [۱۱] .

[۴] norm
[۵] open ball
[۶] closed ball
[۷] doubling space
[۸] doubling dimension
[۹] centerpoint



شکل ۲ــ۱: هر بازه به طول r در فضای یک بعدی را می‌توان با دو بازه به طول $\frac{r}{۲}$ پوشش داد. در نتیجه بعد مضاعف این فضا برابر ۲ است.

تعریف ۷ (kــ مرکز). *(M, d) یک فضای متریک و P مجموعه‌ای از نقاط است. تابع d' را به این صورت تعریف می‌کنیم:*

$$d'(p, S) = \min_{q \in S} d(p, q)$$

هدف ما در مسئله‌ی kــ مرکز، پیدا کردن مجموعه‌ی C است به طوری که $|C| \leqslant k$ و $\max_{p \in P} d(p, C)$ کمینه شود. یعنی:

$$C = \operatorname*{arg\,min}_{C' \subseteq P, |C'| \leqslant k} \max_{p \in P} d(p, C')$$

و شعاع بهینه برابر است با:

$$R_{\mathrm{OPT}} = \min_{C' \subseteq P, |C'| \leqslant k} \max_{p \in P} d(p, C')$$

می‌توان مسئله‌ی kــ مرکز را با توجه به این که علاوه بر مراکز، آیا شعاع بهینه را هم می‌خواهیم پیدا کنیم یا نه به دو مسئله‌ی kــ مرکز با شعاع و kــ مرکز بدون شعاع دسته بندی کرد. در ادامه‌ی این پایان نامه، در صورتی که صراحتاً ذکر نشده باشد، منظورمان از kــ مرکز، نسخه‌ی بدون شعاع مسئله است. برای این که تفاوت این دو مسئله روشن شود، مسئله‌ی ۱ــ مرکز را در نظر بگیرید. برای نسخه‌ی بدون شعاع این مسئله، برای ارائه دادن یک مرکز با ضریب تقریب ۲، کافی است اوّلین نقطه‌ی ورودی را به عنوان مرکز برگردانیم. در حالی که در نسخه‌ی با شعاع این مسئله، بدون بررسی همه‌ی نقاط ورودی ارائه‌ی هیچ تقریبی امکان پذیر نیست.

تعریف ۸ (kــ مرکز با نقاط پرت). *این مسئله مشابه مسئله‌ی kــ مرکز است با این تفاوت که می‌توانیم حداکثر z نقطه را پوشش ندهیم. به طور دقیق تر شعاع بهینه به این صورت تعریف می‌شود:*

$$R_{\mathrm{OPT}} = \min_{Z \subseteq P, |Z| \leqslant z} \min_{C \subseteq P, |C| \leqslant k} \max_{p \in P \setminus Z} d(p, C)$$

مسائل kـ میانه[۱۰] و kـ میانگین نیز شباهت زیادی به مسئله‌ی kـ مرکز دارند. در مسئله‌ی kـ مرکز، هدف خوشه بندی نقطه‌ها است طوری که فاصله‌ی مرکز هر خوشه تا دورترین نقطه‌ی آن خوشه کمینه شود. در مسئله‌ی kـ میانه، هدف این است که مجموع فاصله‌های نقاط هر خوشه تا مرکز آن خوشه کمینه شود و در مسئله‌ی kـ میانگین، هدف کمینه کردن مجموع توان دوم این مقادیر است.

در این پایان نامه گاهی از مسئله‌ی kـ مرکز با نام خوشه بندی مرکزی[۱۱] نیز یاد خواهیم کرد. دلیل این نام گذاری، وجود یک مسئله‌ی خوشه بندی دیگر است که با مسئله‌ی kـ مرکز رابطه‌ی نزدیکی دارد [۱۲].

تعریف ۹ (خوشه بندی دوتایی[۱۲]). *(M, d) یک فضای متریک و P مجموعه‌ای از نقاط است. خوشه بندی دوتایی بهینه برای نقاط P، آن خوشه بندی است که بیشترین قطر خوشه‌ها را کمینه کند.*

می‌توان نشان داد که جواب بهینه برای خوشه بندی مرکزی حداکثر برابر جواب بهینه برای خوشه بندی شعاعی و حداقل نصف آن می‌باشد.

با گسترش شبکه‌ی وب و روزمره شدن فناوری‌های جدید مثل تلفن‌های همراه، دسترسی به حجم بزرگی از اطلاعات وجود دارد. حجم این داده‌ها به قدری زیاد است که بسیاری از الگوریتم‌های قدیمی با دسترسی تصادفی[۱۳] به داده‌ها کارایی قابل قبولی ندارند. این موضوع باعث معرفی مدل جویبار داده[۱۴] شد [۱۳] . در این مدل، داده‌ها یکی یکی به ما داده می‌شوند و ما به داده‌های قدیمی دسترسی نداریم (مگر این که صراحتاً آن‌ها را ذخیره کرده باشیم). در ساده ترین مدل جویبار داده، هدف این است که با حافظه‌ی زیر خطی بتوانیم با یک بار عبور از روی داده‌ها، مسئله را حل کنیم. هدف اصلی ما این است که حافظه‌ی مورد استفاده $O(\log m + \log n)$ باشد که n تعداد داده‌های موجود در جویبار داده و m تعداد حالت‌هایی که هر عنصر جویبار می‌تواند به خود بگیرد است. البته رسیدن به این حافظه برای همه‌ی مسائل امکان پذیر نیست.

این مدل از دو جهت سودمند است. یکی این که می‌توان داده‌های جویبار را بر روی یک دستگاه ذخیره سازی خارجی[۱۵] مثل دیسک سخت[۱۶] تصور کرد و حافظه‌ی محدود مورد استفاده توسط الگوریتم

[۱۰] median
[۱۱] center clustering
[۱۲] pairwise clustering
[۱۳] random access
[۱۴] data stream
[۱۵] external storage device
[۱۶] hard disk

را حافظه‌ی اصلی کامپیوتر تصور کرد. به این ترتیب، می‌توان با یکبار خواندن پیوسته‌ی داده از روی دستگاه ذخیره سازی خارجی، الگوریتم جویبار داده را اجرا کرد که از نظر تعداد ورودی[۱۷] و خروجی[۱۸] به دیسک سخت بهینه است. از طرف دیگر بسیاری از مسائل دنیای واقعی به طور ذاتی ساختار جویبار داده‌ای دارند. برای مثال یک مسیریاب[۱۹] شبکه[۲۰] را در نظر بگیرید که می‌خواهد داده‌های آماری مختلفی را راجع به بسته‌هایی که مسیریابی کرده است نگه داری کند. معمولاً حافظه‌ی مسیریاب‌های شبکه نسبت به تعداد بسته‌هایی که مسیریابی می‌کنند بسیار کوچک است و در نتیجه مسیریاب نمی‌تواند تمام بسته‌هایی را که به آن وارد شده ذخیره کند. از طرفی اگر بسته‌ای فراموش شود، دسترسی دوباره به آن امکان پذیر نیست. در نتیجه، مسیریاب ناچار به استفاده از یک الگوریتم جویبار داده است.

یک گونه‌ی دیگر از الگوریتم‌های جویبار داده، مدل صندوق پول[۲۱] است. در این مدل، هر عنصر از جویبار به همراه یک عدد صحیح مثبت است که تعداد دفعات تکرار آن عنصر را بیان می‌کند.

می‌توان فرض کرد که الگوریتم ما توانایی چند بار عبور از روی داده‌ها را دارد. به این نوع الگوریتم‌ها، الگوریتم جویبار داده‌ی چندگذری[۲۲] می‌گوییم.

تعریف ۱۰ (تابع هزینه‌ی یکنوا[۲۳]). *تابع C، تابع هزینه‌ی یکنوا گفته می‌شود اگر داشته باشیم:*

$$\forall P \subseteq M, Q \subseteq P, x \in M : C_Q(x) \leqslant C_P(x)$$

برای مثال قطر[۲۴] یک مجموعه‌ی نقطه، یک تابع هزینه‌ی یکنوا است چون اضافه کردن نقاط جدید فقط می‌تواند قطر را افزایش دهد.

تعریف ۱۱ (مجموعه‌ی هسته[۲۵]). *برای عدد حقیقی $\alpha \geqslant 1$ و تابع هزینه‌ی C می‌گوییم Q یک α-مجموعه‌ی هسته برای نقاط P است اگر $Q \subseteq P$ و داشته باشیم:*

$$\forall T \subseteq M, \forall x \in M : C_{Q \cup T}(x) \leqslant C_{P \cup T}(x) \leqslant \alpha C_{Q \cup T}(x)$$

[۱۷] input
[۱۸] output
[۱۹] router
[۲۰] network
[۲۱] cash register
[۲۲] multipass
[۲۳] monotone cost function
[۲۴] diameter
[۲۵] core-set

برای مثال می‌توان نشان داد که نگه داشتن دورترین نقاط در $\Theta(\frac{1}{\sqrt{\epsilon}})$ جهت هم زاویه در فضای اقلیدسی دو بعدی برای مسئله‌ی قطر یک ϵـ مجموعه‌ی هسته می‌سازند [۱۴].

در بسیاری از کاربردهای واقعی، داده‌های اخیر برای ما اهمیت بیشتری از داده‌های قدیمی دارند. برای مثال در یک مسیریاب شبکه، احتمالاً بسته‌هایی که هفته‌ها پیش راه یابی شده‌اند به اندازه‌ی آخرین بسته‌ای که راه یابی شده است اهمیت ندارند. برای مدل کردن این واقعیت، روش های مختلفی ارائه شده است. برای مثال [۱۵] روش پیر شدن[۲۶] را معرفی کرد. در این روش، به داده‌های قدیمی به طور نمایی[۲۷] وزن کم‌تری داده می‌شود. به طور دقیق تر، اگر جدید ترین داده را a^{0} و داده‌ی قبلی را a^{-1} و به همین ترتیب داده‌ی i تا قبلی را a^{-i} بنامیم، هدف پیدا کردن مقدار زیر است:

$$\lambda a^{0} + \lambda(1-\lambda)a^{-1} + \lambda(1-\lambda)^{2}a^{-2} + \dots$$

یکی از مشکلات این روش این است که داده‌های قدیمی همچنان بر روی جواب نهایی تاثیر (هر چند اندک) می‌گذارند و در نتیجه به سادگی نمی‌توان آن‌ها را فراموش کرد. همچنین برای بعضی مسائل مانند kـ مرکز که هزینه‌ی ما برابر بیشترین فاصله از مرکز است، بدیهی نیست که چگونه پیر شدن را اعمال کنیم. برای حل این مشکلات، مدل پنجره‌ی لغزان[۲۸] ارائه شد [۱۶] . در این مدل، هدف ما پیدا کردن جواب برای n داده‌ی آخر از جویبار است (به این n داده، پنجره می‌گوییم). به عبارت دیگر داده‌هایی که عمر آن‌ها از n بیشتر است نباید بتوانند بر روی جواب تاثیر بگذارند. همانند مدل جویبار داده، فرض می‌کنیم n عدد بزرگی است و در نتیجه حافظه‌ی ما برای ذخیره‌ی همه‌ی پنجره کافی نیست و در نتیجه باید مسئله را در $o(n)$ حل کنیم.

برای مقایسه‌ی مدل جویبار داده با مدل‌های قبلی، مسئله‌ی ساده‌ی شمارش را معرفی می‌کنیم.

تعریف ۱۲ (مسئله‌ی شمارش). *یک دنباله از داده‌های ۰ و ۱ داده شده است. تعداد ۱ ها را بشمارید.*

بدیهی است که در مدل جویبار داده این مسئله به راحتی قابل حل است (کافی است که تعداد ۱ ها را با استفاده از یک شمارنده‌ی $\log n$ بیتی بشماریم). همچنین در مدل پیر شدن، کافی است که با اضافه

[۲۶] aging
[۲۷] exponential
[۲۸] sliding window

شدن هر نقطه‌ی جدید، مقدار جواب را با استفاده از رابطه‌ی زیر حساب کنیم:

$$f_{i+1} = \lambda d_i + (1 - \lambda) f_i$$

حال، مسئله را در مدل پنجره‌ی لغزان در نظر بگیرید. فرض کنید که n داده‌ی اول به طور تصادفی انتخاب می‌شوند و n داده‌ی دوم همگی ۰ هستند. به طور شهودی می‌توان دید که دنباله‌ی جواب‌های پنجره برای همه‌ی 2^n حالت مختلف n داده‌ی اول متفاوت است و در نتیجه هر الگوریتمی که برای این مسئله بتواند جواب دقیق را محاسبه کند، باید بتواند بین این 2^n حالت تمایز قائل شود و در نتیجه حداقل به

$$\log 2^n = n = \Omega(n)$$

بیت حافظه نیاز دارد. در نتیجه این مسئله به طور دقیق در مدل پنجره‌ی لغزان (با حافظه‌ی $o(n)$) قابل حل نیست.

یکی از روش‌هایی که برای حل مسائل جویبار داده و پنجره‌ی لغزان به کار می‌رود، موازی سازی[۲۹] است. فرض کنید که الگوریتمی داریم که یک تقریب از جواب بهینه را می‌گیرد و یا یک جواب تولید می‌کند و یا شکست می‌خورد و این خاصیت‌ها را دارد:

۱. اگر الگوریتم موفق شود، جوابی به اندازه‌ی حداکثر α برابر تقریب گرفته شده تولید می‌کند.

۲. اگر تقریب داده شده از جواب بهینه بزرگ‌تر باشد، الگوریتم حتماً موفق می‌شود.

همچنین فرض کنید یک کران پایین L و یک کران بالا U بر روی جواب بهینه داریم. حال، بازه‌ی بین L و U را به $O(\log_{1+\epsilon} \frac{U}{L})$ قسمت تقسیم می‌کنیم طوری که نسبت هر تقریب به تقریب قبل، حداکثر $1 + \epsilon$ شود و الگوریتم را همزمان برای همه‌ی تقریب‌ها اجرا می‌کنیم. یعنی هر نقطه که داده می‌شود را به طور جداگانه به همه‌ی نسخه‌های الگوریتم می‌دهیم. می‌دانیم که نسبت یکی از تقریب‌ها به جواب بهینه حداکثر $1 + \epsilon$ است و طبق شرایط بالا، الگوریتمی که با این تقریب اجرا شده موفق می‌شود و جوابی با ضریب تقریب $\alpha(1 + \epsilon)$ تولید می‌کند. در نتیجه الگوریتم کلی به این صورت می‌شود که به ازای همه‌ی تقریب‌ها یک نسخه از الگوریتم را اجرا می‌کنیم و در هر مرحله، جواب اوّلین نسخه از الگوریتم را که موفق شده است به عنوان جواب نهایی باز می‌گردانیم.

[۲۹] parallelization

فصل ۳

کارهای پیشین

۳ـ۱ k ـ مرکز

یکی از اوّلین کارها بر روی این مسئله توسط گنزالز [۵] انجام شد. او در این مقاله یک الگوریتم با ضریب تقریب ۲ برای این مسئله ارائه داد. این الگوریتم به طور کلی به این صورت عمل می‌کند: ابتدا یک نقطه‌ی دلخواه به عنوان یک مرکز انتخاب می‌شود. سپس دورترین نقطه نسبت به مرکزهای انتخاب شده (که در حال حاضر فقط شامل مرکز اوّل است) به عنوان مرکز دوم انتخاب می‌شود و به همین ترتیب الگوریتم در هر مرحله، دورترین نقطه نسبت به مراکز فعلی را به عنوان مرکز بعدی انتخاب می‌کند، تا زمانی که تعداد مراکز به k برسد. به طور شهودی می‌توان دید که این الگوریتم درست کار می‌کند زیرا دو حالت داریم: یا هر مرکز انتخاب شده از یک خوشه‌ی بهینه‌ی مجزا است، که در این صورت می‌دانیم فاصله‌ی همه‌ی نقاط هر خوشه تا مرکز انتخاب شده از آن خوشه حداکثر دو برابر شعاع خوشه است. و یا در یک مرحله از یک خوشه دو مرکز انتخاب شده که از آن جایی که فاصله‌ی این دو مرکز حداکثر دو برابر شعاع خوشه‌ی بهینه است، می‌دانیم در این زمان فاصله‌ی همه‌ی نقاط تا نزدیک ترین مرکز حداکثر برابر این مقدار (دو برابر شعاع خوشه‌ی بهینه) می‌باشد. این الگوریتم با زمان $O(nk)$ قابل پیاده سازی است (کافی است در هر مرحله نزدیک ترین مرکز به هر نقطه را نگه داریم و پس از آن برای پیدا کردن دور ترین مرکز، کافی است تنها فاصله‌ی هر نقطه با نزدیک‌ترین مرکز به آن نقطه را حساب کنیم).

گنزالز در همان مقاله ثابت کرد که این مسئله، NP ـ تمام است (حتی زمانی که در فضای اقلیدسی

دو بعدی هستیم). همچنین تقریب زدن این مسئله با ضریب تقریب $2\cos(\frac{\pi}{6} - \epsilon)$ در فضای دو بعدی و با ضریب تقریب $2 - \epsilon$ در فضای ۳ بعدی، NPـ تمام است. هاچبام و شمویس [۶] نیز یک الگوریتم دیگر با ضریب تقریب ۲ ارائه دادند که از آنجایی که در این پایان نامه از ایده هایی مشابه استفاده شده است، کلیات این الگوریتم را شرح می‌دهیم.

فرض کنید که شعاع بهینه را برای مسئله‌ی kـ مرکز می‌دانیم (فرض کنید مقدار این شعاع بهینه، R_{OPT} باشد). یک نقطه را به دلخواه به عنوان مرکز اوّل انتخاب می‌کنیم و تمام نقاطی را که در فاصله‌ی $2R_{\text{OPT}}$ از این نقطه هستند، حذف می‌کنیم. دقت کنید که همه‌ی نقاطی که در جواب بهینه با مرکز اوّل در یک خوشه قرار می‌گرفته‌اند حذف شده اند (چون فاصله‌ی هر دو با مرکز بهینه حداکثر R_{OPT} بوده است و در نتیجه فاصله‌ی آن‌ها با هم حداکثر $2R_{\text{OPT}}$ است). و در نتیجه با تکرار این فرآیند، از هر خوشه‌ی بهینه، حداکثر یک نقطه انتخاب می‌شود و در نتیجه در پایان (زمانی که همه‌ی نقاط حذف شده اند) حداکثر k مرکز را به عنوان جواب انتخاب کرده ایم.

تا به حال فرض کرده بودیم که شعاع بهینه (R_{OPT}) را می‌دانیم. برای برداشتن این فرض، کافی است به این نکته توجه کنیم که جواب بهینه الزاماً فاصله‌ی بین دو نقطه است (چون در غیر این صورت روی دایره‌های بهینه هیچ نقطه‌ی دیگری قرار نمی‌گیرد و در نتیجه می‌توانیم شعاع همه‌ی دایره‌ها را به اندازه‌ی ϵ کاهش دهیم بدون این که نقطه‌ای از دایره بیرون افتد). در نتیجه کافی است بر روی $\binom{n}{2} = O(n^2)$ فاصله‌ی ممکن بین نقاط یک جستجوی دودویی انجام دهیم و اوّلین جایی که موفق به تولید جواب با کمتر از k مرکز شدیم، الگوریتم را متوقف کنیم.

فدر و گرین [۱۷] نشان دادند که در فضای اقلیدسی با زمان چند جمله‌ای ضریب تقریبی بهتر از ۱/۸۲ نمی‌توان ارائه داد (با فرض $P \neq NP$) و همچنین ارائه‌ی هر ضریب تقریب به $O(n \log k)$ زمان نیاز دارد.

۳ـ۲ kـ مرکز با نقاط پرت

چاریکار و دیگران [۷] مسئله‌ی kـ مرکز با نقاط پرت را بررسی کردند و با کاهش از مسئله‌ی پوشش بیشینه، نشان دادند که تقریب زدن مسئله با مراکز منع شده با ضریب تقریب بهتر از ۳، NPـ تمام است. در این مسئله، مراکز منع شده زیر مجموعه‌ای از نقاط هستند که نباید به عنوان مرکز انتخاب شوند. آن‌ها

همچنین یک الگوریتم با ضریب تقریب ۳ برای این مسئله ارائه دادند که به این صورت کار می‌کند:

۱. مانند قبل فرض می‌کنیم شعاع بهینه (R) را داریم.

۲. بر روی هر نقطه‌ی ورودی یک توپ با شعاع R و یک توپ با شعاع $3R$ در نظر می‌گیریم. (نام آن ها را برای نقطه‌ی iام به ترتیب G_i و E_i می‌گذاریم.)

۳. تا زمانی که کم‌تر از k مرکز داریم، نقطه‌ی j که در G_j آن بیشترین تعداد نقطه وجود دارد را به عنوان مرکز بعدی انتخاب می‌کنیم و همه‌ی نقاط موجود در E_j را از همه‌ی G_i و E_i ها حذف می‌کنیم.

قضیه ۱. *تعداد نقاط پوشانده شده توسط الگوریتم بالا حداقل به اندازه تعداد نقاط پوشانده شده در جواب بهینه است.*

اثبات. فرض کنید جواب بهینه $O = \{O_1, O_2, ..., O_k\}$ باشد که O_i توپی با شعاع R است. به صورت استقرایی نشان می‌دهیم که می‌توان O را طوری مرتب کرد که به ازای هر i ، تعداد نقاط پوشانده شده توسط $E'_i = E_1 \cup E_2 \cup ... \cup E_i$ بیشتر از $O'_i = O_1 \cup O_2 \cup ... \cup O_i$ باشد. این کار را با تصویر کردن هر نقطه در O'_i به یک نقطه‌ی یکتا در E'_i انجام می‌دهیم. فرض کنید در مرحله‌ی i ام الگوریتم، نقطه‌ی i به عنوان مرکز انتخاب شده است. اگر G_i با یکی از $O_i, O_{i+1}, ..., O_k$ اشتراک داشته باشد (مثلاً O_j)، آنگاه O_i را برابر O_j قرار می‌دهیم و همه‌ی نقاط O_j را به خودشان تصویر می‌کنیم (چون می‌دانیم که اگر O_i با G_i اشتراک داشته باشد، آنگاه کاملاً با E_i پوشیده می‌شود و در نتیجه همه‌ی نقاط پوشیده شده توسط O_i با E_i هم پوشیده می‌شوند). در غیر این صورت (یعنی وقتی که G_i با هیچکدام از توپ‌های $O_i, O_{i+1}, ..., O_k$ اشتراک ندارد) می‌دانیم که تعداد نقاط جدید پوشش داده شده توسط G_i از همه‌ی O_j های بعدی بیشتر است (چون در غیر این صورت، O_j به عنوان G_i انتخاب می‌شد.) و در نتیجه همه‌ی نقاط جدید پوشیده شده توسط O_i را به نقاط جدید پوشش داده شده توسط G_i می‌توان تصویر کرد. در ادامه‌ی الگوریتم هیچ نقطه‌ی دیگری نمی‌تواند به این نقاط تصویر شود. چون طبق فرض هیچ کدام از O_j های بعدی با G_i اشتراک نداشتند و در نتیجه در ادامه هیچ کدام از نقاط G_i به خودشان تصویر نمی‌شوند. همچنین هیچ نقطه‌ی دیگری هم به این نقاط تصویر نمی‌شود چون نقاط بعدی به نقاط جدید پوشیده شده توسط یک توپ دیگر مثل G_l تصویر می‌شوند. □

تا به این‌جای کار فرض کرده بودیم که شعاع بهینه را می‌دانیم. برای برداشتن این فرض کافی است

بر روی $O(n^۲)$ فاصله‌ی ممکن بین دو نقطه، جست‌وجوی دودویی انجام دهیم و جواب کوچک ترین شعاعی که با آن موفق به پوشش تمام نقاط به جز حداکثر z تا می‌شویم را به عنوان جواب نهایی برگردانیم.

۳ــ۳ kــ مرکز در مدل جویبار داده

چاریکار و دیگران [۱۸] برای مسئله‌ی خوشه بندی شعاعی یک الگوریتم افزایشی[۱] با ضریب تقریب ۸ ارائه دادند. این مسئله شباهت زیادی به مسئله‌ی kــ مرکز دارد با این تفاوت که هدف ما به جای کمینه کردن فاصله‌ی مرکز با نقاط خوشه‌ها، کمینه کردن شعاع خوشه‌ها است. این الگوریتم مراکز جدید را به جواب قبلی اضافه می‌کند به این صورت که در هر زمان، الگوریتم حداکثر k مرکز را در حافظه نگه می‌دارد که فاصله‌ی بین هر دو مرکز حداقل $۲R$ است و R یک کران پایین برای جواب بهینه (R_{OPT}) است. اگر مرکز جدیدی اضافه شد و در شعاع $۲R$ هیچ کدام از مراکز فعلی نبود، می‌دانیم که R نمی‌تواند کران پایین بر روی جواب بهینه باشد چون $k+۱$ مرکز داریم که فاصله‌ی دوبه‌دوی آن‌ها حداقل $۲R$ است و در نتیجه تنها برای پوشش دادن این نقاط، حداقل شعاع $۲R$ لازم است و در نتیجه می‌توان کران پایین را به $۲R$ افزایش داد.

در این الگوریتم، از نزدیک ترین فاصله بین $k+۱$ نقطه‌ی اوّل، به عنوان کران پایین اوّل استفاده می‌شود (نام این کران را $R.$ می‌گذاریم). نکته‌ی قابل توجه در این الگوریتم این است که اگر در انتخاب کران اوّل خوش شانس باشیم، ممکن است که آخرین کران ما $\epsilon + \frac{R_{\mathrm{OPT}}}{۲}$ باشد. در این صورت، شعاع جواب ما $۸(\frac{R_{\mathrm{OPT}}}{۲}+\epsilon) = ۴R_{\mathrm{OPT}} + ۸\epsilon$ می‌شود. با استفاده از این ایده یک نسخه‌ی تصادفی از این الگوریتم ارائه شده است که امید ریاضی ضریب تقریب آن $۲e$ است.

مککاتچن و خولر [۸] این الگوریتم را بر روی مسئله‌ی kــ مرکز در مدل جویبار داده اعمال کردند. ایده‌ی جدید اصلی این الگوریتم این است که تعداد زیادی نسخه از الگوریتم را با تقریب‌های اوّلیه‌ی متفاوت به صورت موازی اجرا می‌کند طوری که مطمئن می‌شود که همواره خوش شانس می‌شویم (یعنی در مرحله‌ی آخر یکی از الگوریتم‌ها، تقریب برابر $\epsilon + \frac{R_{\mathrm{OPT}}}{۲}$ می‌شود). و در نتیجه ضریب تقریب الگوریتم برابر $۲+\epsilon$ می‌شود. همچنین در این مقاله ایده‌ی این الگوریتم را بر حالتی که نقاط پرت هم داریم اعمال کردند و یک الگوریتم با ضریب تقریب $۴+\epsilon$ به دست آوردند. این الگوریتم $O(\epsilon^{-۱}kz)$ نقطه در حافظه نگه می‌دارد و پیچیدگی زمانی آن $O(\epsilon^{-۱}(kzn + (kz)^۲ \log P))$ است که در این جا P نسبت شعاع بهینه

[۱] incremental algorithm

و نزدیک ترین دو نقطه در ورودی است. از آن جایی که در اکثر الگوریتم‌های ارائه شده در این پایان‌نامه، فرض می‌کنیم که یک کران پایین و بالا بر روی جواب بهینه داریم (و در نتیجه با اعمال موازی سازی، می‌توانیم فرض کنیم که شعاع بهینه را با ضریب $\epsilon+۱$ داریم) و از آنجایی که چالش اصلی این الگوریتم نداشتن شعاع بهینه است، الگوریتم را به طور کامل شرح نمی‌دهیم و فقط نسخه‌ی ساده شده‌ی آن را با فرض این که شعاع بهینه داریم بررسی می‌کنیم.

فرض کنید شعاع بهینه (R_{OPT}) را می‌دانیم. هنگامی که یک نقطه‌ی جدید وارد می‌شود، اگر در شعاع $۴R_{\text{OPT}}$ یکی از مراکز فعلی بود، آن نقطه را فراموش می‌کنیم. در غیر این صورت این نقطه را به لیست نقطه‌های پرت اضافه می‌کنیم. هنگامی که اندازه‌ی لیست نقطه‌های پرت به $kz + z + ۱$ رسید، می‌دانیم که در یکی از خوشه‌های بهینه حداقل $z+۱$ نقطه وجود دارد و در نتیجه از بین نقطه‌های موجود، یکی باید بتواند حداقل $z+۱$ نقطه را با شعاع $۲R_{\text{OPT}}$ بپوشاند. چنین نقطه‌ای را پیدا می‌کنیم و به لیست مراکز اضافه می‌کنیم. همچنین تمام نقاطی را که در شعاع $۴R_{\text{OPT}}$ از این نقطه قرار داشتند را حذف می‌کنیم. در شعاع $۲R_{\text{OPT}}$ این نقطه $z + ۱$ نقطه وجود داشت و می‌دانیم که حداکثر z نقطه‌ی پرت داریم و در نتیجه حداقل یکی از این $z + ۱$ نقطه در یکی از خوشه‌های بهینه قرار دارد و در نتیجه مرکز انتخاب شده تمام این خوشه را با شعاع $۲R_{\text{OPT}} + ۲R_{\text{OPT}} = ۴R_{\text{OPT}}$ پوشش می‌دهد. در نتیجه تمام نقاطی که در این خوشه هستند در این مرحله حذف می‌شوند و در ادامه هم اگر نقطه‌ای در این خوشه باشد، چون در شعاع $۴R_{\text{OPT}}$ از این مرکز است حذف می‌شود. در نتیجه از هر خوشه‌ی بهینه حداکثر یک نقطه انتخاب می‌شود. در این مقاله همچنین یک الگوریتم با ضریب $۳ + \epsilon$ برای حالتی که یک پیشگوی پیداکننده‌ی مرکز[۲] داریم، ارائه شده است. پیشگوی پیداکننده‌ی مرکز، یک الگوریتم است که با داشتن مجموعه‌ی نقاط P و یک عدد j و شعاع x، یک نقطه را پیدا می‌کند که با شعاع x حداقل j نقطه از P را بپوشاند و یا اعلام می‌کند که چنین نقطه‌ای وجود ندارد. در ادامه نشان می‌دهیم که چگونه با استفاده از پیشگوی پیداکننده‌ی مرکز، می‌توان ضریب تقریب الگوریتم ساده شده‌ی بالا را به ۳ کاهش داد. کافی است هنگامی که اندازه‌ی لیست نقاط پرت به $kz + z + ۱$ رسید پیشگوی پیدا کننده‌ی مرکز را با نقاط این لیست و $j = z + ۱$ و $x = R_{\text{OPT}}$ فراخوانی کنیم. می‌دانیم که الزاماً نقطه‌ای وجود دارد که بتواند حداقل $z+۱$ نقطه را با شعاع R_{OPT} بپوشاند (چون در یکی از خوشه‌های بهینه حداقل $z + ۱$ نقطه وجود دارد) و در نتیجه پیشگوی ما موفق می‌شود. ادامه‌ی تحلیل الگوریتم همانند قبل است چون می‌دانیم که حداقل یک نقطه‌ی غیر پرت پوشش داده‌شده است، می‌دانیم که همه‌ی آن خوشه، با شعاع

[۲] center finding oracle

حداکثر $R_{\text{OPT}} + ۲R_{\text{OPT}}$ پوشش داده می‌شوند.

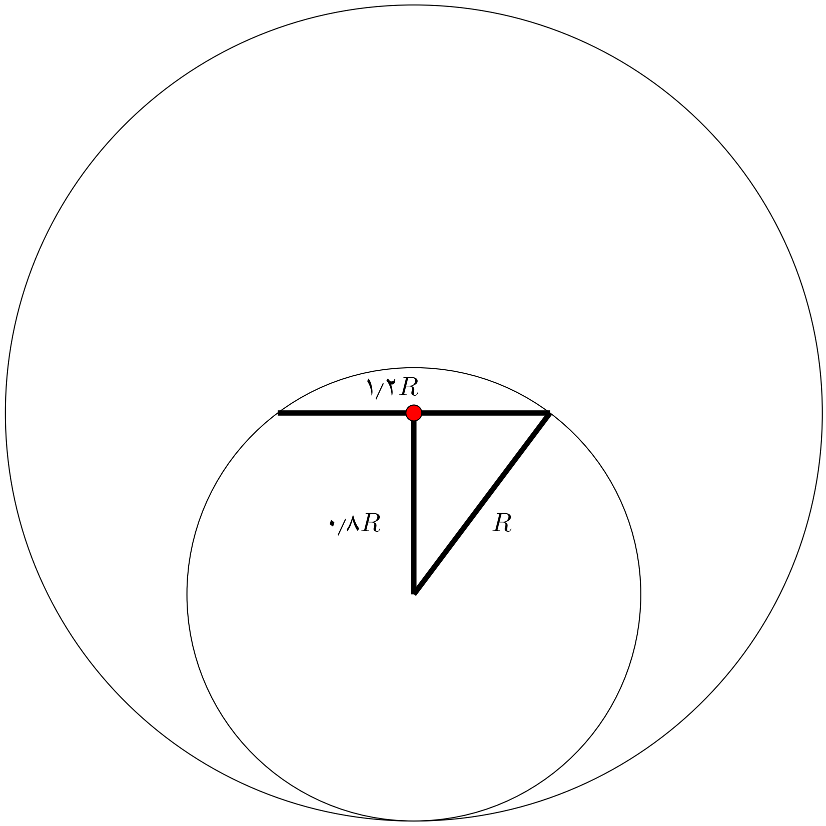
گوها [۱۹] برای مسئله‌ی kـ مرکز در مدل جویبار داده با استفاده از روشی مشابه، به ضریب تقریب $۲ + \epsilon$ دست یافت و همچنین نشان داد که هر الگوریتم برای این مسئله که مراکز بهینه را پیدا کند و همچنین تقریبی بر روی شعاع بهینه ارائه دهد، به $O(\frac{k}{\epsilon})$ فضا نیاز دارد.

ضرابی زاده [۲۰] نشان داد که چگونه در فضای اقلیدسی با ابعاد کوچک یک ϵـ هسته‌ی مرکزی نگه داریم. به طور کلی فرض می‌کنیم که یک کران بالا بر روی جواب بهینه داریم (چنین جوابی می‌تواند به طور موازی با یکی از الگوریتم‌های بالا محاسبه شود). فضای اقلیدسی را طوری شبکه بندی می‌کنیم که قطر هر خانه از شبکه، کمتر از ϵR_{OPT} شود و از این به بعد از هر خانه از شبکه تنها یک نقطه را نگه می‌داریم. می‌توان نشان داد که برای چنین شبکه بندی تنها به $O(\frac{k}{\epsilon^d})$ خانه نیاز داریم و برای به روز رسانی این شبکه بندی زمان لازم $O(n + \frac{k}{\epsilon^d})$ است.

برای وقتی که تعداد مرکزها کم است، کیم و اهن [۲۱] یک الگوریتم با ضریب تقریب $۱/۸ + \epsilon$ ارائه دادند که حافظه‌ی مورد نیاز آن $O(۲^k(k + ۳)!\frac{d}{\epsilon})$ و زمان به روز رسانی آن $O(۲^k(k + ۳)!\frac{d}{\epsilon})$ است. ایده‌ی کلی این الگوریتم برای ۲ مرکز این است که نقطه‌ی اوّل را به عنوان مرکز در نظر می‌گیرد و با آمدن نقطه‌های بعدی، تمام حالت‌های ممکن را در نظر می‌گیرد. مثلاً هنگامی که با اولین نقطه که در شعاع نقطه‌ی اول نبود مواجه شدیم، دو حالت امکان پذیر است: یا این دو نقطه در یک مرکز بهینه وجود دارند، که در این صورت نقطه‌ی وسط پاره‌خط واصل این دو نقطه به عنوان مرکز جدید معرفی می‌شود، و یا این دو نقطه در دو مرکز بهینه‌ی متفاوت قرار دارند که در این صورت نقطه‌ی دوم به عنوان مرکز بعدی اضافه می‌شود. از آنجایی که تعداد حالت‌های ممکن محدود است (چون فقط دو مرکز داریم)، می‌دانیم که الزاماً یکی از فرض‌های ما درست بوده است. حال نسخه‌ای از الگوریتم را در نظر بگیرید که با تقریب شعاع $۱/۲R_{OPT}$ اجرا شده است. اگر در فرض ما هیچ‌گاه نقطه‌ی وسطی دو مرکز به عنوان مرکز جدید انتخاب نشده باشد، می‌دانیم که همه‌ی نقاط در شعاع $۱/۲R_{OPT}$ از مرکزهای انتخابی هستند. در غیر این صورت، می‌توان نشان داد که همه‌ی نقاط آن مرکز با شعاع $۱/۸R_{OPT}$ توسط نقطه‌ی میانی دو مرکز پوشانده می‌شوند (شکل ۳ـ۱).

حاتمی و ضرابی زاده [۲۲] این الگوریتم را برای ۲ـ مرکز در حالتی که نقاط پرت داریم گسترش دادند و به یک الگوریتم با ضریب تقریب $۱/۸ + \epsilon$ رسیدند که به $O(dz^۲(d^۲ + \frac{z^۲}{\epsilon}))$ حافظه نیاز دارد.

برای مسئله‌ی ۱ـ مرکز در فضای اقلیدسی و در حالتی که مجاز به استفاده از نقطه‌های جدید باشیم،



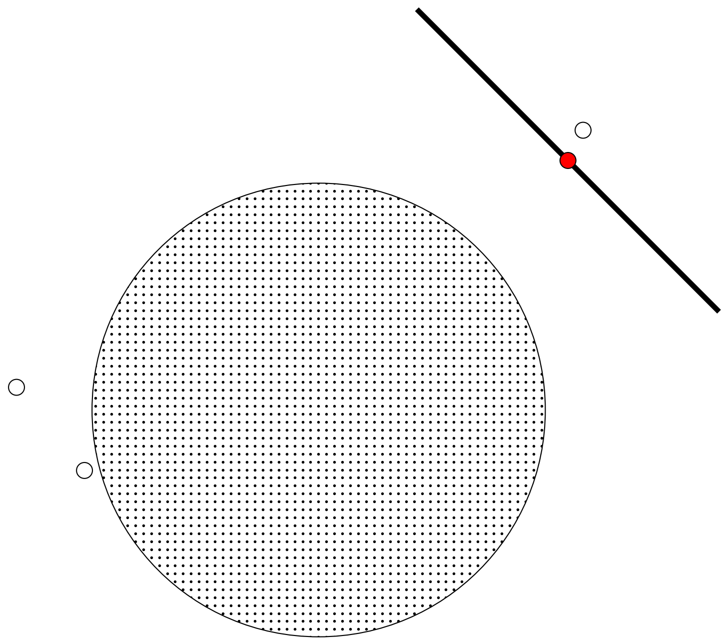
شکل ۳ـ۱: نقطه‌ی میان دو مرکز، همه‌ی دایره را با شعاع $۱/۸R$ می‌پوشاند.

اگروال و شاراسکومار [۲۳] یک الگوریتم با ضریب تقریب $\frac{1+\sqrt{3}}{2}$ ارائه داند. چن و پاتاک [۲۴] با یک تحلیل بهتر نشان دادند که ضریب تقریب همان الگوریتم کمتر از ۱/۲۲ است.

برای ۱ـ مرکز در حالتی که نقاط پرت داریم، ضرابی زاده و موخوپادهیای [۲۵] نشان دادند که نقطه‌ی میانی یک مجموعه نقطه با اندازه‌ی $(z+1)(d+1)$ الزاماً داخل دایره‌ی بهینه قرار می‌گیرد (شکل ۳ـ۲). و در نتیجه می‌دانیم نقطه‌ی میانی $(z+1)(d+1)$ نقطه، بقیه‌ی نقاط توپ بهینه را با شعاع حداکثر $2R_{OPT}$ می‌پوشاند. همچنین با ایده‌ی خارج کردن نزدیک ترین نقطه به نقطه‌ی میانی و اضافه کردن آن نقطه به یک الگوریتم تقریب ۱ـ مرکز بدون نقاط پرت، به یک الگوریتم با ضریب تقریب $\sqrt{2}\alpha$ دست یافتند که α ضریب تقریب الگوریتم بدون نقاط پرت است. در نتیجه با استفاده از از الگوریتم با ضریب ۱/۲۲ برای این مسئله، می‌توان به ضریب تقریب $1/73 = 1/22\sqrt{2}$ دست یافت.

برای محاسبه‌ی دقیق نقطه‌ی میانی، چن [۲۶] یک الگوریتم با زمان $O(n^{d-1})$ ارائه داد. این الگوریتم در بعد فضای اقلیدسی نمایی است و در نتیجه برای بعدهای بالا کاربرد ندارد. دقت کنید که در الگوریتم بالا، نیازی به استفاده از نقطه‌ی میانی دقیق نیست و اگر تقریبی هم برای نقطه‌ی میانی وجود داشته باشد قابل استفاده است. کلارکسون و سایرین [۲۷] چنین الگوریتمی برای تقریب نقطه‌ی میانی ارائه دادند. الگوریتم آن‌ها یک $\Theta(\frac{1}{d^2})$ـ نقطه‌ی میانی را با احتمال حداقل $1-\delta$ در زمان $d^9 \log d + d^8 \log \frac{1}{\delta} + d^{5+\epsilon} \log^2 \frac{1}{\delta}$ محاسبه می‌کند. با استفاده از این الگوریتم، می‌توان تقریب بالا را در زمان چندجمله‌ی در d و n محاسبه کرد.

سسارلو و سایرین [۲۸] ، مسئله‌ی kـ مرکز با نقاط پرت را در فضای متریک با بعد مضاعف محدود بررسی کردند. ایده‌ی اصلی مقاله‌ی آن‌ها این بود که برای حل مسئله‌ی kـ مرکز، ابتدا یک الگوریتم تقریبی برای k'ـ مرکز اجرا می‌کردند و مقدار $k' > k$ طوری انتخاب می‌شد که جواب یک الگوریتم برون خط بر روی این k' مرکز، تقریب خوبی بر روی جواب اصلی مسئله باشد. با استفاده از این ایده، توانستند به ضریب تقریب $3+\epsilon$ دست بیابند. حافظه‌ی مورد نیاز این الگوریتم، $O((k+z)(\frac{1}{\epsilon})^D)$ است که D بعد مضاعف فضا می‌باشد. ورگگاری [۲۹] تقریب‌هایی برای بعد مضاعف فضاهای اقلیدسی به دست آورد و به طور کلی ثابت کرد که بعد مضاعف فضای n بعدی از $n\sqrt{n}2^n$ کوچک تر است.



شکل ۳ـ۲: از روی هر نقطه بیرون دایره‌ی بهینه، حداقل یک خط می‌گذرد که دایره‌ی بهینه در یک سمت آن قرار دارد و در نتیجه در سمت دیگر آن، کمتر از $z + 1$ نقطه قرار دارد

۳ـ۴ پنجره‌ی لغزان

مدل پنجره‌ی لغزان توسط داتار و سایرین [۱۶] معرفی شد. همان طور که در مقدمه اشاره کردیم، حل دقیق مسئله‌ی شمارش در مدل پنجره‌ی لغزان امکان پذیر نیست. در این مقاله یک الگوریتم با ضریب تقریب $\epsilon + 1$ برای این مسئله ارائه شد که به $O(\frac{1}{\epsilon}\log^{2} n)$ بیت حافظه نیاز دارد و همچنین ثابت شد که این الگوریتم از نظر حافظه بهینه است. ایده‌ی کلی این الگوریتم این است که داده‌ها را به دسته‌هایی به اندازه‌ی $1, 2, 4, \dots, 2^{i}, \dots$ دسته بندی می‌کند و به ازای هر دسته، تنها زمان ورود جدیدترین عضو را نگه می‌دارد. هنگامی که تعداد دسته‌ها با یک اندازه به یک حد معین رسید($\left\lceil \frac{1}{2\epsilon} \right\rceil$)، قدیمی‌ترین دو دسته با هم ترکیب می‌شوند و یک دسته با اندازه‌ی بیشتر تولید می‌کنند. برای ارائه‌ی تقریب از تعداد ۱ها در پنجره، تعداد عناصر همه‌ی دسته‌ها به جز دسته‌ی آخر را با هم جمع می‌کنیم (می‌دانیم که همه‌ی این نقاط همچنان در پنجره هستند) در مورد دسته‌ی آخر، نمی‌دانیم که چه تعداد از نقاط این دسته همچنان در پنجره هستند در نتیجه اگر تعداد نقاط داخل این دسته برابر C باشد، عدد به دست آمده را با $\frac{C}{2}$ جمع می‌زنیم. در نتیجه می‌دانیم که تقریب ارائه شده با مقدار واقعی جواب حداکثر به اندازه‌ی $\frac{C}{2}$ فاصله دارد و از آنجایی که حداقل $\left\lceil \frac{1}{2\epsilon} \right\rceil$ دسته با اندازه‌ی $1, \dots, \frac{C}{4}, \frac{C}{2}$ داریم، می‌دانیم که خطای نسبی جواب،

حداکثر به اندازه‌های $(\epsilon+1)$ است. دقت کنید که جواب ارائه شده توسط این الگوریتم می‌تواند از مقدار واقعی بیش‌تر یا کم‌تر باشد. در ادامه‌ی این پایان نامه به الگوریتمی نیاز خواهیم داشت که جواب آن از مقدار واقعی فقط کم‌تر باشد. به راحتی می‌توان نشان داد که اگر در الگوریتم بالا، تعداد دسته‌ها را برابر $\lceil \frac{1}{\epsilon} \rceil$ قرار دهیم و دسته‌ی آخر را در تقریب ارائه شده توسط الگوریتم در نظر نگیریم، جواب به دست آمده حداکثر با ضریب $\epsilon + 1$ از جواب واقعی کم‌تر است و هیچ وقت از جواب واقعی بیش‌تر نیست. همچنین حافظه‌ی مورد نیاز برای این الگوریتم برابر $O(\frac{1}{\epsilon}\log^2 n)$ است. در این مقاله همچنین مسائلی مانند به دست آوردن جمع داده‌های پنجره و داده‌هایی آماری بر روی پنجره بررسی شد.

چن و سجاد [۳۰] ، برای مسئله‌ی قطر در پنجره‌ی لغزان در فضای اقلیدسی d بعدی الگوریتمی با ضریب تقریب $\epsilon + 1$ ارائه دادند. روش کلی این است که ابتدا یک داده ساختار برای تقریب زدن بیشینه در پنجره‌ی لغزان ارائه دادند. سپس با استفاده از آن، قطر داده‌ها را برای فضای ۱ بعدی تخمین زدند. سپس برای تقریب فضای d بعدی، مسئله‌ی ۱ بعدی را بر روی تعداد زیادی خط در جهات مختلف حل کردند و بیشبنه‌ی جواب‌ها را به عنوان جواب نهایی بازگرداندند. در کل حافظه‌ی مورد نیاز این الگوریتم، $O((\frac{1}{\epsilon})^{(d+1)/2}\log\frac{\alpha}{\epsilon})$ است که R در اینجا نسبت قطر به نزدیک ترین دو نقطه در پنجره است.

کوهن اداد و سایرین [۹] برای مسئله‌ی kــ مرکز در مدل پنجره‌ی لغزان یک الگوریتم با ضریب تقریب ۶ ارائه دادند. این الگوریتم از تکنیک موازی سازی استفاده می‌کند و به ازای هر تقریب، یا یک دسته بندی با شعاع حداکثر ۶ برابر آن تقریب ارائه می‌دهد و یا ثابت می کند که با تقریب داده شده، دسته بندی امکان پذیر نیست. فرض کنید تقریب داده شده به الگوریتم، R است. این الگوریتم مجموعه‌ای از مرکزها را نگه می‌دارد و به ازای هر مرکز، آخرین نقطه‌ای که در شعاع $2R$ از آن مرکز قرار دارد را نگه می‌دارد (بقیه‌ی نقاطی که در شعاع $2R$ آن مرکز هستند فراموش می‌شوند) در نهایت برای جواب دادن به یک پرس و جو، سعی می‌کنیم نقاط موجود در حافظه را با شعاع $2R$ بپوشانیم. می‌دانیم که در شعاع $4R$ هر نقطه‌ی فراموش شده، یک نقطه در حافظه وجود دارد. در نتیجه در کل شعاع مرکزهای پیدا شده توسط الگوریتم، حداکثر برابر $2R + 4R = 6R$ می‌شود. تعداد نقاط ذخیره شده در حافظه توسط این الگوریتم، $O(\frac{k\log\alpha}{\epsilon})$ که در اینجا α نسبت پخشش[۳] نقاط است (همان α در الگوریتم چن). در این مقاله همچنین نشان داده شد که برای ارائه‌ی ضریب تقریب بهتر از ۴، حداقل به ذخیره‌ی $\Omega(\sqrt[3]{n})$ نقطه نیازمندیم.

[۳] spread

۳ـ۵ قطر

اگر فرض کنیم که می‌توان فاصله‌ی دو نقطه را در $O(1)$ محاسبه کرد، الگوریتمی بدیهی برای محاسبه‌ی قطر n نقطه در $O(n^2)$ وجود دارد: فاصله‌ی بین همه‌ی $\binom{n}{2} = O(n^2)$ جفت نقطه را محاسبه می‌کنیم و کمینه‌ی این مقادیر را باز می‌گردانیم. می‌توان نشان داد که محاسبه‌ی قطر در فضای اقلیدسی d بعدی حداقل به $\Omega(n \log n)$ زمان نیاز دارد [۳۱]. در فضای دو بعدی رسیدن به این کران پایین ساده است. می‌دانیم که دو سر قطر الزاماً بر روی پوسته‌ی محدب قرار دارند. در نتیجه کافی است پوسته‌ی محدب نقاط را در $O(n \log n)$ محاسبه کنیم [۳۲] و سپس به ازای هر نقطه روی پوسته‌ی محدب، با استفاده از جست‌وجوی دودویی در $O(\log n)$ دور ترین نقطه به آن را پیدا کنیم. در فضای اقلیدسی سه بعدی کلارکسون و شور یک الگوریتم تصادفی[۴] با زمان $O(n \log n)$ ارائه دادند [۳۳]. راموس اوّلین الگوریتم قطعی[۵] را برای محاسبه‌ی قطر در فضای اقلیدسی سه بعدی ارائه داد [۳۴]. در فضاهای متریک، الگوریتم ساده‌ای برای تقریب زدن قطر با ضریب تقریب ۲ وجود دارد. کافی است یک نقطه را به دلخواه انتخاب کنیم و دورترین نقطه نسبت به آن را بیابیم. می‌دانیم که فاصله‌ی هر نقطه تا یکی از دو سر قطر حداقل نصف طول قطر است. در فضای اقلیدسی، [۳۵] نشان داد که الگوریتم ساده‌ی ۱ برای نقاط S جوابی با ضریب تقریب $\sqrt{3}$ باز می‌گرداند.

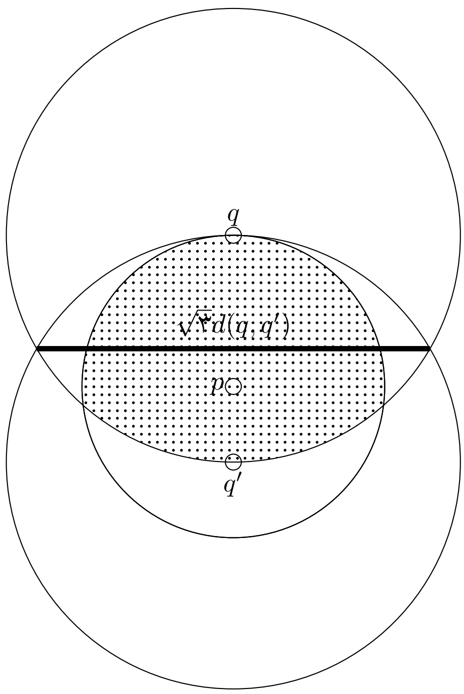
الگوریتم ۱ تقریب قطر

۱: $p = $ یک نقطه‌ی دلخواه در S
۲: $q = \arg\max_{t \in S} d(t, p)$
۳: $q' = \arg\max_{t \in S} d(t, q)$
۴: برگردان $d(q, q')$

همان طور که در شکل ۳ـ۳ مشخص است، تمام نقاط در ناحیه‌ی هاشور خورده قرار دارند و قطر ناحیه‌ی هاشور خورده حداکثر برابر $\sqrt{3} d(q, q')$ است. چن [۳۶] نشان داد که می‌توان در زمان $O(n + \frac{1}{\epsilon^{1/5(d-1)}})$ جوابی با ضریب تقریب $1 + \epsilon$ به دست آورد.

در مدل پنجره‌ی لغزان و در فضای متریک، چن و سجاد نشان دادند که می‌توان با نگه داشتن $O(\sqrt{n} \log_{1+\epsilon} \alpha)$ نقطه، به ضریب تقریب $6 + \epsilon$ دست یافت. کوهن‌ـاداد و سایرین [۹] نشان دادند که اگر مقدار α را بدانیم می‌توان جوابی با ضریب تقریب $3 + \epsilon$ را با ذخیره‌ی $\frac{1}{\epsilon} \log \alpha$ نقطه به دست آورد (الگوریتم چن و سجاد نیازی به دانستن α نداشت).

[۴] randomized
[۵] deterministic



شکل ۳ـ۳: تمام نقاط در ناحیه‌ی هاشور خورده قرار دارند

فصل ۴

نتایج جدید

۴ـ۱ kـ مرکز بدون شعاع

در این بخش ابتدا چند الگوریتم ساده برای مسئله‌ی kـ مرکز بدون شعاع در مدل پنجره‌ی لغزان مطرح می‌کنیم.

۴ـ۱ـ۱ ۱ـ مرکز اقلیدسی

برای مسئله‌ی اقلیدسی بدون شعاع، مشاهده می‌کنیم که می‌توان با استفاده از روش ضرابی زاده و موخوپادهیای [۲۵] به یک الگوریتم با ضریب تقریب ۲ دست یافت. فرض کنید الگوریتمی داریم که یک βـ نقطه‌ی میانی را برای مجموعه‌ای از نقاط محاسبه می‌کند. الگوریتم ما $\beta(z+1)$ نقطه‌ی اخیر را در حافظه نگه می‌دارد (فرض می‌کنیم که اندازه‌ی پنجره از $\beta(z+1)$ بیشتر است. در غیر این صورت، می‌توانیم همه‌ی نقاط موجود در پنجره را در حافظه نگه داریم و مسئله‌ی برون خط را بر روی آن‌ها حل کنیم). سپس برای هر پرس‌وجو، الگوریتم پیدا کردن βـ نقطه‌ی میانی را بر روی این نقاط اجرا می‌کنیم و مرکز به دست آمده توسط این الگوریتم را به عنوان مرکز پنجره‌ی فعلی باز می‌گردانیم. در ادامه ادعا می‌کنیم که نقطه‌ی میانی پیدا شده توسط الگوریتم، باید داخل کره‌ی بهینه باشد و در نتیجه همه‌ی نقاط کره را با شعاع حداکثر دو برابر شعاع بهینه می‌پوشاند.

لم ۲. *مرکز به دست آمده در خط ۱۰ داخل دایره‌ی بهینه قرار می‌گیرد.*

الگوریتم ۲ ۱ـ مرکز اقلیدسی با نقاط پرت

۱: $L = \emptyset$
۲: **رویه** اضافه(p)
۳: $L = L \cup \{p\}$
۴: اگر $|L| > \beta(z+1)$:
۵: قدیمی ترین نقطه‌ی L را حذف کن
۶: **رویه** پرسش
۷: اگر $|L| < \beta(z+1)$:
۸: با استفاده از الگوریتم برون خط، جواب را برای نقاط L به دست بیاور.
۹: در غیر این صورت:
۱۰: یک βـ نقطه‌ی میانی برای نقاط L به دست بیاور و به عنوان مرکز بازگردان

اثبات. می‌دانیم که از هر نقطه خارج از ابرکره، یک ابرصفحه می‌گذرد به طوری که ابرکره در یک سمت ابرصفحه قرار می‌گیرد (برای مثال ابرصفحه‌ی عمود بر خط واصل نقطه و مرکز کره). از آنجایی که حداکثر z نقطه بیرون کره وجود دارد، می‌دانیم که در سمت دیگر صفحه، حداکثر z نقطه وجود دارد. از طرفی می‌دانیم که در هر سمت هر صفحه‌ی گذرنده از یک βـ نقطه‌ی مرکز، حداقل $\frac{1}{\beta}$ از نقاط وجود دارند و از آن جایی که $\beta(z+1)$ نقطه داریم، می‌دانیم که در هر سمت هر صفحه‌ی گذرنده از نقطه‌ی بازگردانده شده توسط خط ۱۰، حداقل $\frac{1}{\beta}(\beta(z+1)) = z+1$ نقطه وجود دارد. در نتیجه این نقطه نمی‌تواند بیرون دایره باشد. □

لم ۳. *الگوریتم ۲ مرکزی باز می‌گرداند که همه‌ی نقاط پنجره را با شعاع حداکثر دو برابر شعاع بهینه پوشش می‌دهد.*

اثبات. اگر جواب با خط ۸ بازگردانده شود، می‌دانیم که همه‌ی نقاط پنجره را داریم و در نتیجه می‌توانیم تک تک نقاط موجود در حافظه را به عنوان مرکز چک کنیم و آن را که با کمترین شعاع می‌تواند همه‌ی نقاط به جز z تا را پوشش دهد به عنوان مرکز بهینه بازگردانیم (در این حالت جواب بهینه را باز می‌گردانیم). در غیر این صورت طبق لم ۲ می‌دانیم که مرکز بازگردانده شده توسط خط ۱۰، داخل دایره‌ی بهینه است و در نتیجه می‌تواند همه‌ی نقاط دایره‌ی بهینه را حدکثر با دو برابر شعاع بهینه پوشش دهد. □

لم ۴. *اگر یک الگوریتم برای پیدا کردن βـ نقطه‌ی مرکز داشته باشیم که در $O(f(n))$ بتواند نقطه‌ی مرکزی را پیدا کند، می‌توان الگوریتم ۲ را با حافظه‌ی $O(\beta z)$ و زمان درج $O(1)$ و زمان پرس‌وجوی*

$O(f(\beta(z+1))) + (\beta z)^2)$ *پیاده سازی کرد.*

اثبات. تنها حافظه‌ی مورد نیاز الگوریتم، $\beta(z+1)$ نقطه‌ی اخیر است که به $O(\beta z)$ حافظه نیاز دارد. برای درج، کافی است نقطه‌ی جدید را به لیست نقاط اضافه کنیم و در صورت لزوم، قدیمی‌ترین نقطه را حذف کنیم. با استفاده از یک لیست پیوندی، این کار به سادگی در $O(1)$ امکان پذیر است. خط ۸ را می‌توان به سادگی در $O((\beta z)^2$ پیاده سازی کرد. به ازای هر کدام از $\beta(z+1)$ مرکز ممکن، شعاعی را که این مرکز می‌تواند حداقل $z+1$ نقطه‌ی دیگر را پوشش دهد به دست می‌آوریم. برای این کار، کافی است که $z+1$امین دورترین نقطه به آن مرکز را به دست آوریم که با استفاده از الگوریتم انتخاب، در $O(\beta z)$ قابل پیاده سازی است [۳۷]. طبق فرض، خط ۱۰ را می‌توان در $O(f(\beta(z+1)))$ محاسبه کرد. در نتیجه به طور کلی زمان درج $O(f(\beta(z+1)) + (\beta z)^2)$ است. □

در قضیه‌ی زیر و همچنین در ادامه‌ی این پایان نامه، حافظه‌ی مورد نیاز برای الگوریتم‌ها را بر اساس تعداد نقاط مشخص می‌کنیم. همچنین فرض می‌کنیم که طول عمر هر نقطه در $O(1)$ از روی هر نقطه قابل محاسبه است. در نتیجه بدیهی است که هر نقطه حداقل به $\Omega(\log k + \log n)$ حافظه نیاز دارد ($\log k$ برای این که هر نقطه‌های بین مرکزهای مختلف باید از یکدیگر قابل تمایز باشند و $\log n$ چون در هر نقطه، طول عمر آن نقطه نیز ذخیره شده است).

قضیه ۵. *برای مسئله‌ی ۱ــ مرکز اقلیدسی با نقاط پرت در مدل پنجره‌ی لغزان، الگوریتمی با ضریب تقریب ۲ و حافظه‌ی $O(d^2 z)$ و زمان درج $O(1)$ و زمان پرس‌وجوی*

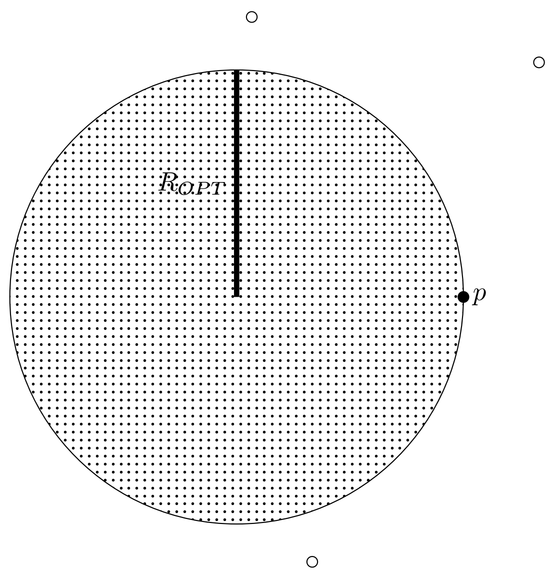
$$O(d^9 \log d + d^8 \log \frac{1}{\delta} + d^{5+\epsilon} \log^2 \frac{1}{\delta})$$

وجود دارد که با احتمال $1-\delta$ موفق می‌شود.

اثبات. با استفاده از الگوریتم پیدا کردن نقطه‌ی میانی تقریبی کلارکسون [۲۷]، می‌توانیم یک $\frac{1}{3e(\frac{d+2}{2})}$ـ مرکز میانی را با احتمال حداقل $1-\delta$ در زمان $O(d^9 \log d + d^8 \log \frac{1}{\delta} + d^{5+\epsilon} \log^2 \frac{1}{\delta})$ بیابیم. در نتیجه با قرار دادن این تابع در لم ۴، به کران‌های ذکر شده می‌رسیم. □

شکل ۴ــ۱ یک مثال تنگ[1] برای الگوریتم ۲ نشان می‌دهد. در این شکل، فرض کنید دایره‌ی بزرگ

[1] tight example



شکل ۴ــ۱: مثال تنگ برای الگوریتم ۲

نشان داده شده، توپ بهینه باشد و دایره‌های توخالی، نقاط پرت و دایره‌های توپر، نقاط موجود در حافظه باشند. در این مثال، همه‌ی نقاط موجود در حافظه روی نقطه‌ی p هستند. در نتیجه نقطه‌ی میانی این نقاط، نقطه‌ی p خواهد بود که دایره‌ی بهینه را با شعاع $۲R_{OPT}$ می‌پوشاند.

۴ــ۱ــ۲ ۱ــمرکز متریک

برای مسئله‌ی ۱ــمرکز متریک بدون شعاع با نقاط پرت در مدل پنجره‌ی لغزان یک الگوریتم با ضریب تقریب ۴ ارائه می‌دهیم. الگوریتم به این صورت کار می‌کند: $۲z + ۱$ نقطه‌ی اخیر را در حافظه نگه می‌داریم. می‌دانیم که از بین این نقاط، حداقل $۲z + ۱ - z = z + ۱$ تا نقطه‌ی پرت نیستند و در نتیجه در یک دایره بهینه با شعاع R_{OPT} قرار دارند. از بین نقاط داخل حافظه، آن را پیدا می‌کنیم که حداقل $z + ۱$ نقطه‌ی دیگر را با کم‌ترین شعاع ممکن پوشش دهد. می‌دانیم که این شعاع از $۲R_{\text{OPT}}$ کم تر است چون $z + ۱$ نقطه‌ای که در داخل دایره‌ی بهینه هستند، می‌توانند با شعاع $۲R_{\text{OPT}}$ بقیه‌ی نقاط داخل دایره‌ی بهینه را پوشش دهند. در نتیجه نقطه‌ی بازگردانده شده، می‌تواند با شعاع حداکثر $۲R_{\text{OPT}}$، حداقل $z + ۱$ نقطه را پوشش دهد. می‌دانیم که از این $z + ۱$ نقطه، حداقل یکی نقطه‌ی پرت نیست (چون حداکثر z نقطه‌ی پرت داریم). در نتیجه مرکز بازگردانده شده، با شعاع $۲R_{\text{OPT}}$ حداقل یک نقطه‌ی غیر پرت را می‌پوشاند که خود، فاصله‌اش با سایر نقاط غیر پرت حداکثر $۲R_{\text{OPT}}$ است و در نتیجه مرکز بازگردانده

شده، همه‌ی نقاط غیر پرت را با شعاع حداکثر $2R_{\mathrm{OPT}} + 2R_{\mathrm{OPT}} = 4R_{\mathrm{OPT}}$ می‌پوشاند.

الگوریتم ۳ ۱ـ مرکز متریک با نقاط پرت

۱: $L = \emptyset$
۲: **رویه** اضافه(p)
۳: $L = L \cup \{p\}$
۴: اگر $|L| > 2z + 1$:
۵: قدیمی ترین نقطه‌ی L را حذف کن
۶: **رویه** پرسش
۷: از بین نقاط موجود در حافظه، آن را که با کم‌ترین شعاع همه‌ی نقاط به جز حداکثر z تا را پوشش می‌دهد بازگردان

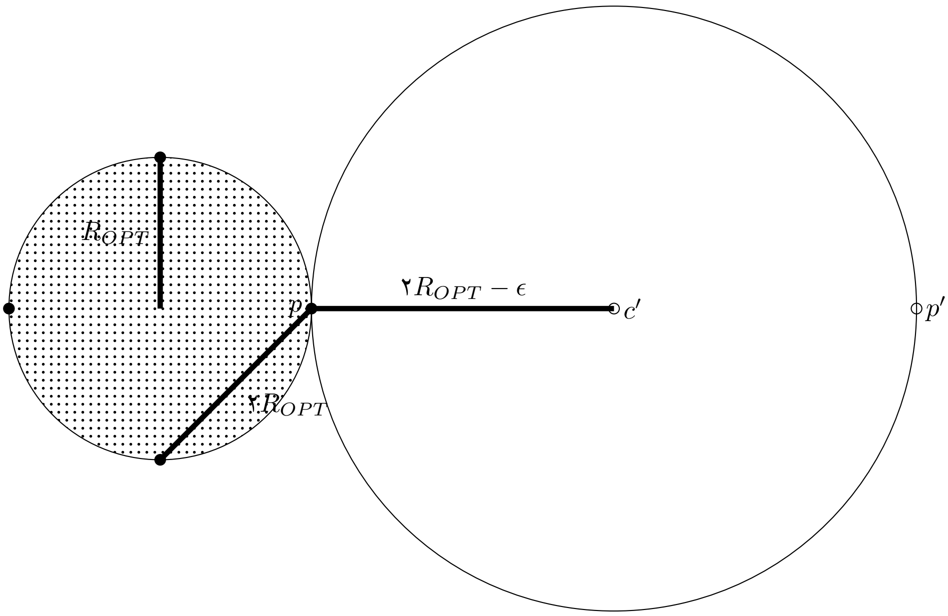
لم ۶. *اگر $|L| = 2z + 1$، مرکز بازگردانده شده در خط ۷ حداقل $z + 1$ نقطه را با شعاع $2R_{\mathrm{OPT}}$ پوشش می‌دهد.*

اثبات. از بین $2z + 1$ نقطه‌ی موجود در حافظه، حداکثر z تای آن‌ها پرت هستند و در نتیجه حداقل $z + 1$ تای آن‌ها پرت نیستند. این $z + 1$ نقطه می‌توانند همدیگر را با شعاع $2R_{\mathrm{OPT}}$ پوشش دهند چون فاصله‌ی هر کدام تا مرکز بهینه حداکثر R_{OPT} است و طبق نامساوی مثلثی در فضاهای متریک، می‌دانیم که فاصله‌ی آن‌ها با یکدیگر حداکثر $R_{\mathrm{OPT}} + R_{\mathrm{OPT}} = 2R_{\mathrm{OPT}}$ است. در نتیجه از آن‌جایی که نقطه‌ای را بر می‌گردانیم که با کم‌ترین شعاع $z + 1$ نقطه را پوشش دهد، می‌دانیم که این کم‌ترین شعاع باید از $2R_{\mathrm{OPT}}$ کم‌تر باشد. □

قضیه ۷. *مرکز بازگردانده شده در الگوریتم ۳، همه‌ی نقاط توپ بهینه را با شعاع حداکثر $4R_{\mathrm{OPT}}$ پوشش می‌دهد.*

اثبات. طبق لم ۶، می‌دانیم که مرکز بازگردانده شده، با شعاع $2R_{\mathrm{OPT}}$ حداقل $z + 1$ نقطه را پوشش می‌دهد. در نتیجه با شعاع $2R_{\mathrm{OPT}}$ حداقل یک نقطه‌ی غیر پرت را پوشش می‌دهد چون حداکثر z نقطه‌ی پرت داریم. از طرفی همان طور که قبلاً توضیح دادیم، فاصله‌ی این نقطه‌ی غیر پرت تا سایر نقطه‌های غیر پرت حداکثر $2R_{\mathrm{OPT}}$ است. در نتیجه فاصله‌ی مرکز بازگردانده شده تا همه‌ی نقاط غیر پرت حداکثر $2R_{\mathrm{OPT}} + 2R_{\mathrm{OPT}} = 4R_{\mathrm{OPT}}$ است. □

قضیه ۸. *برای ۱ـ مرکز در فضای متریک، یک الگوریتم با ضریب تقریب ۴ و حافظه‌ی $O(z)$ و زمان درج $O(1)$ و زمان پرس‌وجوی $O(z^2)$ وجود دارد.*



شکل ۴ــ۲: مثال تنگ برای الگوریتم ۲

اثبات. مشابه اثبات قضیه‌ی ۵. □

شکل ۴ــ۲، یک مثال تنگ برای الگوریتم ۳ نشان می‌دهد. در این شکل، دایره‌ی هاشور خورده، دایره‌ی بهینه است و $z + ۱$ نقطه‌ی سیاه داخل آن هستند که فاصله‌ی دو به دوی آن‌ها با هم $۲R_{\mathrm{OPT}}$ است. یکی از نقاط پرت c' است که همه‌ی z نقطه‌ی پرت و یک نقطه‌ی غیر پرت را (در مجموع $z + ۱$ نقطه) با شعاع $۲R_{\mathrm{OPT}} - \epsilon$ پوشش می‌دهد و در نتیجه این نقطه به عنوان نقطه‌ای که با کم‌ترین شعاع $z + ۱$ نقطه را پوشش می‌دهد انتخاب می‌شود. همان طور که از روی شکل مشخص است، فاصله‌ی این نقطه تا دورترین نقطه در دایره‌ی بهینه برابر $۴R_{\mathrm{OPT}} - \epsilon$ است.

۴ــ۲ ۱ــ مرکز در فضای متریک با بعد مضاعف ثابت

اگر بدانیم که بعد مضاعف فضای متریک ما ثابت است، با یک تغییر ساده در الگوریتم ۳ می توان به ضریب تقریب $۲ + \epsilon$ رسید. می‌دانیم که دایره‌ی بهینه با شعاع R_{OPT} را می‌توان با تعدادی دایره با شعاع $\frac{\epsilon}{۲}R_{\mathrm{OPT}}$ پوشاند. در نتیجه به تعدادی نقطه ذخیره می‌کنیم که مطمئن شویم داخل یکی از این دایره‌ها، حداقل $z + ۱$ نقطه وجود دارد و در نتیجه می‌دانیم کمترین شعاعی که با آن حداقل $z + ۱$ نقطه قابل

پوشش هستند، حداکثر برابر $2(\frac{\epsilon}{4}R_{\mathrm{OPT}}) = \epsilon R_{\mathrm{OPT}}$ است و در نتیجه با این شعاع حداقل یک نقطه‌ی غیر پرت را پوشش می‌دهیم و همه‌ی نقاط غیر پرت را با شعاع $2R_{\mathrm{OPT}} + \epsilon R_{\mathrm{OPT}} = (2+\epsilon)R_{\mathrm{OPT}}$ می‌توانیم پوشش دهیم.

الگوریتم ۴ ۱ ـ مرکز در فضای متریک با بعد مضاعف ثابت

۱: $L = \emptyset$
۲: **رویه** اضافه(p)
۳: $L = L \cup \{p\}$
۴: اگر $|L| > z + (z+1)D^{\lceil \log_2 \frac{2}{\epsilon} \rceil}$:
۵: قدیمی ترین نقطه‌ی L را حذف کن
۶: **رویه** پرسش
۷: از بین نقاط موجود در حافظه، آن را که با کم‌ترین شعاع، حداقل $z+1$ نقطه را پوشش می‌دهد بازگردان

قضیه ۹. *در یک فضای متریک با بعد مضاعف D، یک توپ با شعاع R را می‌توان با حداکثر $D^{\lceil \log_2 \frac{1}{\epsilon} \rceil}$ توپ با شعاع ϵR پوشش داد.*

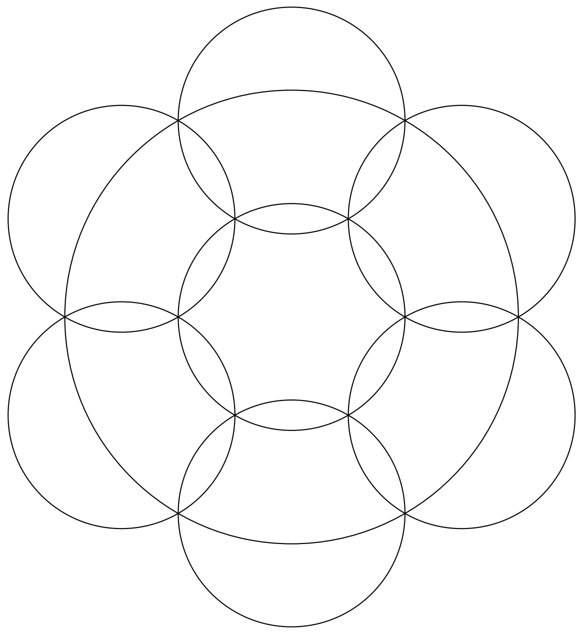
اثبات. یک توپ با شعاع R را می‌توان با D توپ با شعاع $\frac{R}{2}$ پوشاند. به همین ترتیب هر کدام از آن توپ‌ها را می‌توان با D توپ با شعاع $\frac{R}{4}$ پوشاند. پس از $\lceil \log_2 \frac{1}{\epsilon} \rceil$ بار تکرار این فرآیند، به توپ هایی با شعاع کم‌تر از ϵR می‌رسیم که طبق تعریف، کاملاً توپ اصلی را می‌پوشانند. □

لم ۱۰. *اگر $|L| = z + (z+1)D^{\lceil \log_2 \frac{2}{\epsilon} \rceil}$، توپی با شعاع $\frac{\epsilon}{4}R_{OPT}$ وجود دارد که شامل حداقل $z+1$ نقطه است.*

اثبات. می‌دانیم که از بین $|L|$ نقطه، حداقل $|L| - z = (z+1)D^{\lceil \log_2 \frac{2}{\epsilon} \rceil}$ نقطه‌ی غیر پرت داریم که در یک توپ با شعاع R_{OPT} قرار دارند. از طرفی می‌دانیم که این توپ را می‌توان با $D^{\lceil \log_2 \frac{2}{\epsilon} \rceil}$ توپ با شعاع $\frac{\epsilon}{4}R_{\mathrm{OPT}}$ پوشاند. طبق اصل لانه‌ی کبوتری می‌دانیم که یکی از این توپ‌ها باید شامل حداقل $\frac{(z+1)D^{\lceil \log_2 \frac{2}{\epsilon} \rceil}}{D^{\lceil \log_2 \frac{2}{\epsilon} \rceil}} = z+1$ نقطه باشد. □

لم ۱۱. *نقطه‌ی بازگردانده شده در خط ۷ الگوریتم ۴، همه‌ی نقاط غیر پرت را با شعاع حداکثر $(2+\epsilon)R_{\mathrm{OPT}}$ می‌پوشاند.*

اثبات. طبق لم ۱۰، می‌دانیم که توپی با شعاع $\frac{\epsilon}{4}R_{\mathrm{OPT}}$ وجود دارد که حداقل $z+1$ نقطه از نقاط داخل حافظه را پوشش می‌دهد. در نتیجه هر کدام از این نقاط، با شعاع حداکثر $2(\frac{\epsilon}{4}R_{\mathrm{OPT}}) = \epsilon R_{\mathrm{OPT}}$ حداقل



شکل ۴ـ۳: بعد مضاعف فضای دو بعدی برابر ۷ است

$z+1$ نقطه را پوشش می‌دهند و از این $z+1$ نقطه می‌دانیم که حداقل یک نقطه پرت نیست. این نقطه‌ی غیر پرت می‌تواند همه‌ی نقاط غیر پرت را با شعاع حداکثر $2R_{\mathrm{OPT}}$ بپوشاند. در نتیجه در کل حداکثر فاصله‌ی مرکز بازگردانده شده تا نقاط غیر پرت برابر $2R_{\mathrm{OPT}} + \epsilon R_{\mathrm{OPT}} = (2+\epsilon)R_{\mathrm{OPT}}$ است. □

قضیه ۱۲. *برای مسئله‌ی ۱ـ مرکز با نقاط پرت در فضای متریک با بعد مضاعف D، الگوریتمی با ضریب تقریب $2+\epsilon$ و حافظه‌ی $O(zD^{\log_2 \frac{1}{\epsilon}})$ و زمان درج $O(1)$ و زمان پرس‌وجوی $O(z^2D^{2\log_2 \frac{1}{\epsilon}})$ وجود دارد.*

اثبات. مشابه اثبات قضیه‌ی ۵. □

برای مثال، همان طور که در شکل ۴ـ۳ نشان داده شده است، می‌دانیم که بعد مضاعف فضای اقلیدسی دو بعدی برابر ۷ است [۳۸] . در نتیجه برای ارائه‌ی ضریب تقریب $2+\epsilon$ در این فضا، به

$$O(z7^{\log_2 \frac{1}{\epsilon}}) = O(z\frac{1}{\epsilon^{\log_2 7}}) = O(z\frac{1}{\epsilon^{2/81}})$$

حافظه نیاز داریم. در پایان ذکر می‌کنیم که به جای استفاده از بعد مضاعف در فضاهای اقلیدسی، می‌شود از ایده‌ی شبکه بندی صفحه استفاده کرد.

۴ـ۳ الگوریتم با ضریب تقریب $\epsilon + ۳$ برای ۱ـ مرکز در فضای متریک

همان طور که در تحلیل الگوریتم ۳ دیدیم، ضریب تقریب این الگوریتم ۴ است. دلیل این موضوع، این است که می‌دانیم همواره حداقل $z + ۱$ نقطه در حافظه داخل دایره‌ای با شعاع R_{OPT} هستند و در نتیجه می‌توانند همدیگر را با شعاع $۲R_{\mathrm{OPT}}$ پوشش دهند. نکته‌ی جالب در مورد این الگوریتم این است که اگر مرکز بهینه داخل حافظه باشد، می‌دانیم که این مرکز می‌تواند حداقل $z + ۱$ نقطه را با شعاع R_{OPT} پوشش دهد و در نتیجه شعاع کمینه از R_{OPT} کمتر است و در نتیجه ضریب تقریب الگوریتم برابر ۳ می‌شود.

در این قسمت، سعی می‌کنیم الگوریتم ۳ را طوری تغییر دهیم که همیشه مثل حالت بالا خوش‌شانس شویم، یعنی همواره نقطه‌ای با شعاع کوچک‌تر یا مساوی شعاع بهینه در حافظه موجود باشد.

در این الگوریتم، برای سادگی نوشتار، فرض می‌کنیم به همراه هر نقطه، طول عمر باقی مانده‌ی آن نقطه را هم داریم که آن را با ttl نشان می‌دهیم. (بدیهی است که این کمیت با ذخیره‌ی زمان ورود نقطه در $O(۱)$ قابل محاسبه است).

الگوریتم جدید، مانند الگوریتم قبلی $۲z + ۱$ نقطه‌ی اخیر را در حافظه نگه می‌دارد. امّا این بار، هنگامی که قدیمی ترین نقطه در حال خارج شدن از آرایه است، کم‌ترین شعاعی که این نقطه می‌تواند حداقل $z + ۱$ نقطه از نقاط آرایه را پوشش دهد را محاسبه می‌کنیم. دقت کنید که اگر این نقطه‌ی خارج شده، مرکز بهینه‌ی پنجره باشد، این حداقل شعاع حداکثر برابر R_{OPT} خواهد بود. سپس این شعاع را به همراه نقطه‌ی خارج شده و طول عمر آن، در یک داده ساختار تقریب کمینه در پنجره‌ی لغزان قرار می‌دهیم. این داده ساختار یک صف دو طرفه است به این صورت که هر چه از انتهای صف به سمت ابتدای صف حرکت می‌کنیم، شعاع‌ها کم‌تر و عمر نقاط بیش‌تر می‌شود. برای اضافه کردن یک نقطه‌ی جدید، از انتهای صف همه‌ی نقاطی را که شعاعشان از شعاع جدید بیشتر است را از صف خارج می‌کنیم (فراموش کردن این نقاط مشکلی ایجاد نمی‌کند چرا که نقطه‌ای که می‌خواهیم اضافه کنیم هم جدیدتر است و هم شعاع کم‌تری دارد). اگر شعاع نقطه‌ی جدید بیشتر از $\epsilon + ۱$ برابر شعاع قبلی بود، نقطه‌ی جدید را به انتهای صف اضافه می‌کنیم. در غیر این صورت، نقطه‌ی جدید را جایگزین نقطه‌ی پیشین می‌کنیم (بدون تغییر شعاع). برای پرس‌وجو، ابتدا در آرایه نقطه‌ای را پیدا می‌کنیم که با کم‌ترین شعاع، حداقل $z + ۱$ نقطه را پوشش دهد. اگر این شعاع، کم‌تر از شعاع ابتدای صف بود، همین نقطه را باز می‌گردانیم و در غیر این صورت، نقطه‌ی ابتدای صف را باز می‌گردانیم.

الگوریتم ۵ الگوریتم با ضریب تقریب ۳ برای ۱ــ مرکز در فضای متریک

$Q =$ یک صف دو طرفه‌ی خالی
$L = \emptyset$
رویه درج(p)
  تا وقتی ابتدای صف از پنجره خارج شده است:
    $Q.popfront()$
  $L = L \cup \{p\}$
  اگر $|L| > 2z + 1$:
    $p' =$ قدیمی ترین نقطه در L
    $L = L \setminus \{p'\}$
    $R =$ کم ترین شعاعی که p' با آن می‌تواند حداقل $z + 1$ نقطه از L را پوشش دهد
    تا وقتی $Q \neq \emptyset$ و $Q.back.R \geqslant R$:
      $Q.popback()$
    اگر $Q = \emptyset$ یا $Q.back.R < (1 + \epsilon)R$:
      $Q.pushback(\{p : p', R : R, ttl : p'.ttl\})$
    در غیر این صورت:
      $Q.back.p = p'$
      $Q.back.ttl = p'.ttl$ ◁ *دقت کنید که $Q.back.R$ را تغییر نمی‌دهیم*

رویه پرس‌وجو
  $R_1 = \infty$
  اگر $Q \neq \emptyset$:
    $R_1 = Q.front.R$
  $R_2 =$ کم ترین شعاعی که با آن یک نقطه از L می‌تواند حداقل $z + 1$ نقطه‌ی دیگر از L را پوشش دهد (نام این نقطه را p می‌گذاریم)
  اگر $R_1 < R_2$:
    برگردان $Q.front.p$
  در غیر این صورت:
    برگردان p

لم ۱۳. *اگر $|L| < 2z + 1$، تابع پرس‌وجو، مرکز بهینه را باز می‌گرداند.*

اثبات. اگر $|L| < 2z + 1$، می‌دانیم که همه‌ی نقاط پنجره را داریم. در نتیجه با چک کردن تک تک نقاط به عنوان مرکز، می‌توانیم مرکز بهینه را به دست آوریم. □

در نتیجه از این به بعد فرض می‌کنیم که L پر شده است. همچنین فرض می‌کنیم که اندازه‌ی پنجره از $|L|$ بزرگ‌تر است چون در غیر این صورت، می‌توانستیم همه‌ی نقاط پنجره را در حافظه نگه داریم و نیازی به استفاده از این الگوریتم نبود.

لم ۱۴. *در خط ۲۳، کمینه‌ی R_1 و R_2 حداکثر برابر R_{OPT} است.*

اثبات. دو حالت داریم: یا مرکز بهینه هنوز در L است و یا قبلاً از L خارج شده و در Q درج شده است. در حالت اوّل، می‌دانیم که R_2 حداکثر برابر R_{OPT} است. به این دلیل که از آن‌جایی که فرض کردیم که اندازه‌ی پنجره از $2z + 1$ بیش‌تر است، می‌دانیم همه‌ی نقاط موجود در L در پنجره‌ی فعلی زنده هستند. در نتیجه مرکز بهینه (که در L قرار دارد) باید بتواند حداقل $|L| - z = z + 1$ نقطه از L را با شعاع حداکثر R_{OPT} پوشش دهد.

در حالت دوم، می‌دانیم که مرکز بهینه هنگام خروج از L، قادر به پوشش دادن حداقل $z + 1$ نقطه از L با شعاع R_{OPT} بوده است. دلیل این موضوع، مشابه دلیل حالت قبل است. از آن‌جایی که همه‌ی نقاط موجود در L هنگام خروج مرکز بهینه از L از آن مرکز جدیدتر بوده‌اند، می‌دانیم که اگر آن مرکز در پنجره‌ی فعلی زنده باشد، همه‌ی آن نقاط نیز باید زنده باشند. و اگر شعاع بهینه فعلی برابر R_{OPT} باشد، باید حداقل $z + 1$ تا از آن نقاط توسط مرکز بهینه با شعاع R_{OPT} پوشش داده شوند. از آنجایی که این مرکز بهینه در Q در خانه‌ای با شعاع حداکثر R_{OPT} درج شده است، می‌دانیم که R_1 نمی‌تواند از R_{OPT} بیشتر باشد. □

لم ۱۵. *کمینه‌ی واقعی، حداکثر $\epsilon + 1$ برابر کمینه‌ی موجود در Q است.*

اثبات. اگر کمینه‌ی واقعی، هنگام درج به عنوان عنصر جدید وارد شده باشد، می‌دانیم کمینه‌ی موجود در Q برابر با مقدار کمینه‌ی واقعی است. در غیر این صورت، کمینه‌ی واقعی جایگزین عنصر دیگری شده است و طبق خط ۱۳ می‌دانیم مقدار آن حداکثر $\epsilon + 1$ برابر مقدار قبلی موجود در این خانه بوده است. □

قضیه ۱۶. *مرکز ارائه شده توسط الگوریتم ۵، همه‌ی نقاط توپ بهینه را با شعاع حداکثر $(3+\epsilon)R_{\mathrm{OPT}}$ می‌پوشاند.*

اثبات. می‌دانیم که نقطه‌ی ارائه شده، با شعاع حداکثر $(1+\epsilon)\min(R_1, R_2)$ می‌تواند حداقل $z+1$ نقطه را پوشش دهد(لم ۱۵). همچنین بر اساس لم ۱۴ می‌دانیم که $R_{\mathrm{OPT}} \geqslant \min(R_1, R_2)$ در نتیجه مرکز ارائه شده، با شعاع حدکثر $(1+\epsilon)R_{\mathrm{OPT}}$ حداقل $z+1$ نقطه را پوشش می‌دهد که یکی از آن‌ها پرت نیست و در نتیجه در دایره‌ی بهینه است و فاصله‌اش با سایر نقاط داخل دایره‌ی بهینه حداکثر برابر $2R_{\mathrm{OPT}}$ است. در نتیجه مرکز ارائه شده می‌تواند همه‌ی نقاط دایره‌ی بهینه را با شعاع حداکثر $2R_{\mathrm{OPT}} + (1+\epsilon)R_{\mathrm{OPT}}$ بپوشاند. □

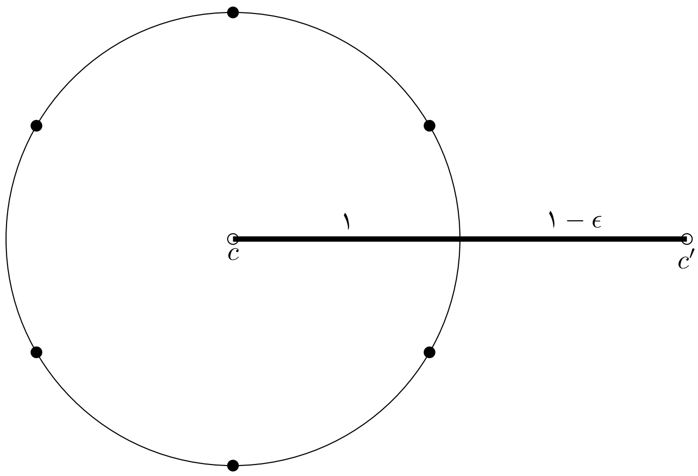
برای تحلیل این الگوریتم، متغیر α را برای نقاط داخل پنجره تعریف می‌کنیم که به نسبت دورترین دو نقطه به نزدیک‌ترین دو نقطه گفته می‌شود.

قضیه ۱۷. *الگوریتمی با ضریب تقریب $3+\epsilon$ مسئله‌ی ۱ــ مرکز با نقاط پرت در مدل پنجره‌ی لغزان با زمان درج سرشکن $O(z)$ و زمان پرس‌وجوی $O(z^2)$ و حافظه‌ی $O(z + \log_{1+\epsilon}\alpha)$ وجود دارد.*

اثبات. در تابع درج، همه‌ی خطوط در $O(1)$ قابل انجام هستند به جز خط‌های ۱۰ و ۱۱. امّا می‌دانیم که هر بار اجرای حلقه در خط ۱۱ معادل یک درج پیشین است. در نتیجه هزینه‌ی سرشکن اجرای این حلقه به ازای هر درج $O(1)$ است. خط ۱۰ با استفاده از الگوریتم انتخاب، به سادگی در $O(z)$ قابل پیاده سازی است. در نتیجه زمان سرشکن کلی این تابع برابر $O(z)$ می‌شود.

در تابع پرس‌وجو، تنها جایی که در $O(1)$ قابل پیاده سازی نیست، پیدا کردن نقطه‌ای در L است که $z+1$ نقطه را با کم‌ترین شعاع بپوشاند. همان طور که قبلاً توضیح داده شد، این مرحله در $O(z^2)$ قابل پیاده سازی است و در نتیجه زمان کلی این تابع برابر $O(z^2)$ است.

الگوریتم ما در دو جا حافظه مصرف می‌کند: L و Q. می‌دانیم که حداکثر اندازه‌ی L برابر $2z+1$ است و در نتیجه حافظه‌ی آن از $O(z)$ است. می‌دانیم که هر عنصر از Q فاصله‌ی دو نقطه را مشخص می‌کند و می‌دانیم نسبت مقدار دو عنصر متوالی در Q حداقل $1+\epsilon$ است. در نتیجه نسبت دورترین دو نقطه به نزدیک‌ترین دو نقطه حداقل برابر $(1+\epsilon)^{|Q|}$ است که می دانیم از α کم‌تر است و در نتیجه $|Q|$ حداکثر برابر $\log_{1+\epsilon}\alpha$ است و حافظه‌ی کلی الگوریتم $O(z + \log_{1+\epsilon}\alpha)$ می‌شود. □



شکل ۴ـ۴: مثال تنگ برای الگوریتم ۵

شکل ۴ـ۴ یک مثال تنگ برای این الگوریتم را نشان می‌دهد. در این مثال، ابتدا نقاط سیاه روی محیط یکی یکی وارد می‌شوند. فاصله‌ی دو به دوی این نقاط ۲ است. بعد از این که حداقل $z + ۱$ تا از این نقاط فراموش شدند (یعنی در هیچ کدام از ساختارهای L یا Q وجود نداشتند)، نقطه‌ی c وارد می‌شود. فاصله‌ی این نقطه تا همه‌ی نقاطی که تا کنون وارد شده است (چه آن‌هایی که در حافظه موجود هستند و چه آن‌هایی که فراموش شده‌اند) برابر ۱ است. سپس نقطه‌ی c' وارد می‌شود. فاصله‌ی این نقطه تا c برابر ۲ و با نقاطی به جز c که در حافظه موجود هستند برابر ϵ − ۱ و با نقاطی که فراموش شده‌اند برابر ϵ − ۳ است. به سادگی می‌توان چک کرد که همه‌ی فاصله‌های بیان شده در نامساوی مثلثی صدق می‌کنند. حال مشاهده می‌کنیم که الگوریتم ما نقطه‌ی c' را به عنوان مرکز بهینه باز می‌گرداند که شعاع بهینه‌ی آن برابر ϵ − ۳ است (چون بیش از z نقطه فراموش شده بودند) در حالی که مرکز بهینه نقطه‌ی c است که با شعاع ۱ می‌تواند همه‌ی نقاط به جز c' را پوشش دهد.

۴ـ۴ موازی سازی

همان طور که در بخش تعریف‌ها اشاره شد، اگر برای یک مسئله الگوریتمی داشته باشیم که با گرفتن یک تقریب، جوابی با مقدار حداکثر α برابر آن تقریب تولید کند و یا شکست بخورد که در این صورت تضمین می‌کند که تقریب داده شده از جواب بهینه کم‌تر است، می‌توانیم با اجرای موازی چند نسخه از این الگوریتم، جوابی با ضریب تقریب $\alpha(۱ + \epsilon)$ پیدا کنیم. فرض کنید نام الگوریتم ما با این دو

خاصیت A باشد و یک کران پایین و بالا بر روی جواب بهینه داشته باشیم که به ترتیب آن‌ها را $R_{\min}$ و $R_{\max}$ می‌نامیم. همچنین تعریف می‌کنیم $\Delta = \frac{R_{\max}}{R_{\min}}$. در ادامه فرض می‌کنیم که $R_{\min} \leqslant \frac{R_{\mathrm{OPT}}}{1+\epsilon}$ و $R_{\max} \geqslant R_{\mathrm{OPT}}(1+\epsilon)$.

الگوریتم ۶ موازی سازی

۱: به ازای i در $\{0, ..., \log_{1+\epsilon} \Delta\}$:
۲: الگوریتم $A(R_{\min}(1+\epsilon)^i)$ را اجرا کن
۳: **رویه** پرس‌وجو
۴: برگردان جواب اوّلین نسخه از الگوریتم‌ها که موفق می‌شود

قضیه ۱۸. *الگوریتم ۶، جوابی با ضریب تقریب $\alpha(1+\epsilon)$ باز می‌گرداند.*

اثبات. می‌دانیم که حداقل یکی از تقریب‌ها مثل $R.$، از R_{OPT} بیشتر است. و طبق فرض، این تقریب حتماً موفق می‌شود و جوابی با شعاع حداکثر $\alpha R.$ بر می‌گرداند. همچنین می‌دانیم که اگر تقریب کوچک‌تری موفق شده باشد، جواب بازگردانده شده توسط آن تقریب، از $\alpha R.$ کوچک تر است. همچنین می‌دانیم که $R.$ حدکثر برابر با $R_{\mathrm{OPT}}(1+\epsilon)$ است. در نتیجه مقدار جواب بازگردانده شده حداکثر $\alpha(1+\epsilon)R_{\mathrm{OPT}}$ است. □

قضیه ۱۹. *اگر زمان درج الگوریتم A برابر $O(I(n))$ و زمان جست‌وجوی آن برابر $O(Q(n))$ و حافظه‌ی آن برابر با $O(S(n))$ باشد، می‌توان با استفاده از موازی سازی الگوریتمی با زمان درج $O(I(n) \log_{1+\epsilon} \Delta)$ و زمان جست‌وجوی $O(Q(n) \log_2 \log_{1+\epsilon} \Delta)$ و حافظه‌ی $O(S(n) \log_{1+\epsilon} \Delta)$ با ضریب تقریب $\alpha(1+\epsilon)$ به دست آورد.*

اثبات. از آن جایی که الگوریتم را $\log_{1+\epsilon} \Delta$ بار اجرا کرده‌ایم، زمان درج و حافظه‌ی الگوریتم در $\log_{1+\epsilon} \Delta$ ضرب می‌شوند.

برای پرس‌وجو، می‌توان از جست‌وجوی دودویی برای پیدا کردن کوچک ترین تقریب موفق استفاده کرد و در نتیجه زمان جست‌وجوی A در $\log_2 \log_{1+\epsilon} \Delta$ ضرب می‌شود. □

۴ـ۵ kـ مرکز در فضای متریک

در این قسمت برای kـ مرکز در فضای متریک کلی یک الگوریتم با ضریب تقریب ۸ ارائه می‌دهیم. برای این کار، از روش موازی سازی استفاده می‌کنیم. به این صورت که یک الگوریتم در پنجره‌ی لغزان ارائه می‌دهیم که با فرض داشتن تقریبی بر جواب بهینه، یا مراکزی بر می‌گرداند که همه‌ی نقاط غیر پرت را با شعاع حداکثر ۸ برابر آن تقریب پوشش دهند و یا اثبات می‌کند که جواب بهینه از تقریب ارائه شده بیشتر است.

فرض کنید تقریبی که به ما داده شده است، R باشد. الگوریتم ما لیستی از نقاط را در مجموعه‌ی C نگه می‌دارد. هنگامی که یک نقطه‌ی جدید وارد می‌شود، اگر در شعاع $۲R$ یکی از نقاط موجود در C بود، به لیست شاهد آن نقطه اضافه می‌شود (برای هر نقطه c موجود در C، $z + ۱$ نقطه‌ی اخیر را که در شعاع $۲R$ آن بوده‌اند را در لیست $c.r$ نگه می‌داریم. نام این لیست نقاط شاهد نقطه‌ی c است). هنگامی که یکی از مراکز موجود در C از پنجره خارج می‌شود، آن را از C حذف می‌کنیم و همه‌ی نقاط شاهد آن را به مجموعه‌ی D اضافه می‌کنیم. همچنین هرگاه نقطه‌ای حذف شود، همه‌ی نقاط قدیمی‌تر از آن را از مجموعه‌ی D حذف می‌کنیم. اگر اندازه‌ی مجموعه‌ی C از $k + z$ بیشتر شد، می‌دانیم که تقریب ارائه شده برای این نقاط درست نبوده است. همچنین تا زمانی که قدیمی‌ترین این نقاط در پنجره وجود دارد، می‌دانیم که جوابی با این تقریب امکان پذیر نیست. در نتیجه می‌توانیم همه‌ی نقاط موجود در D که از این نقطه قدیمی‌تر هستند را حذف کنیم. برای پرس‌وجو اگر اندازه‌ی C از $k + z$ بیشتر باشد، می‌دانیم که جوابی نداریم. در غیر این صورت، یک الگوریتم برون‌ـخط با ضریب تقریب ۴ را بر روی نقاط موجود در حافظه اجرا می‌کنیم. در صورتی که جواب این الگوریتم از $۴R$ بیشتر شد، الگوریتم شکست می‌خورد و در غیر این صورت، نقاط بازگردانده شده توسط الگوریتم برون‌ـخط را به عنوان مرکز باز می‌گردانیم.

لم ۲۰. *اگر R از R_{OPT} بیشتر باشد، الگوریتم ۷ موفق می‌شود.*

اثبات. الگوریتم در دو صورت ممکن است شکست بخورد: وقتی که $|C|$ از $k + z$ بیشتر شود و وقتی که الگوریتم برون‌ـخط موفق نشود نقاط موجود در حافظه را با شعاع $۴R$ پوشش دهد. می‌دانیم که حالت دوم در صورتی که الگوریتم برون خط ما ضریب تقریب ۴ داشته باشد و $R > R_{\mathrm{OPT}}$ غیر ممکن است. اگر حالت اوّل رخ دهد، بیشتر از $k + z$ نقطه داریم که فاصله‌ی دوبه‌دوی آن‌ها از $۲R$ بیشتر است.

الگوریتم ۷ kـ مرکز با نقاط پرت

```
۱: C = ∅
۲: D = ∅
۳: رویه درج(p)
۴:     اگر مرکزی مثل c در C وجود دارد که از پنجره خارج شده است:
۵:         حذف(c)
۶:     اگر نقطه‌ای مثل d در D وجود دارد که از پنجره خارج شده است:
۷:         D = D \ {d}
۸:     cp = {c|c ∈ C, d(c,p) ⩽ ۲R}
۹:     اگر |cp| > ۰:
۱۰:        جدیدترین نقطه در nc = cp
۱۱:        nc.r = nc.r ∪ {p}
۱۲:        اگر |nc.r| > z + ۱:
۱۳:            قدیمی‌ترین نقطه در nc.r را حذف کن
۱۴:    در غیر این صورت:
۱۵:        p.r = ∅
۱۶:        C = C ∪ {p}
۱۷:        اگر |C| > k + z + ۱:
۱۸:            قدیمی‌ترین مرکز در old = C
۱۹:            حذف(old)
۲۰:        اگر |C| > k + z:
۲۱:            قدیمی‌ترین مرکز در old = C
۲۲:            همه‌ی نقطه‌های قدیمی‌تر از old را از D حذف کن
۲۳: رویه حذف(c)
۲۴:     C = C \ {c}
۲۵:     D = D ∪ c.r
۲۶:     همه‌ی نقاط قدیمی‌تر از c را از D حذف کن
۲۷: رویه پرس‌وجو
۲۸:     اگر |C| > k + z:
۲۹:         برگردان شکست
۳۰:     در غیر این صورت:
۳۱:         همه‌ی نقاط موجود در حافظه را با استفاده از الگوریتم برون خط با شعاع حداکثر ۴R بپوشان
۳۲:         اگر خط ۳۱ موفق شد:
۳۳:             برگردان مراکز به دست آمده در خط ۳۱
۳۴:         در غیر این صورت:
۳۵:             برگردان شکست
```

یعنی حداقل $k+۱$ نقطه‌ی غیر پرت داریم که فاصله‌ی دوبه‌دوی آن‌ها حداقل $۲R$ است و در نتیجه هیچ k توپی با شعاع R نمی‌توانند همه‌ی این $k+۱$ را پوشش دهند. در نتیجه جواب بهینه باید از R بیشتر باشد که با فرض در تناقض است. □

لم ۲۱. *اگر الگوریتم ۷ موفق شود، جوابی باز می‌گرداند که همه‌ی نقاط پنجره به جز حداکثر z تا را با شعاع $۸R$ پوشش می‌دهد.*

اثبات. دقت کنید که در طول الگوریتم همه‌ی نقاطی که حذف می‌شوند یا از پنجره خارج شده اند و یا می‌دانیم که تا زمانی که این نقاط زنده هستند، الگوریتم موفق نمی‌شود (چون $|C| > k+z$ است) و در نتیجه حذف کردن این نقاط ضرری ندارد. تنها جایی که این موضوع صحت ندارد، خط ۱۳ است. در این خط، زمانی که تعداد نقاط شاهد یک مرکز از $z+۱$ بیشتر می‌شود، قدیمی‌ترین نقطه‌ی شاهد را حذف می‌کنیم. در نتیجه اگر نقطه‌ای در پایان پنجره زنده باشد ولی در حافظه موجود نباشد و همچنین در پایان این پنجره موفق شویم، می‌دانیم که این نقطه در خط ۱۳ حذف شده است. از طرفی در این صورت، می‌دانیم در پایان پنجره هم آن مرکز $z+۱$ نقطه‌ی شاهد دارد (چون نقاط شاهد تنها زمانی کاهش می‌یابند که از پنجره خارج شوند ولی می‌دانیم که همه‌ی نقاط شاهد زنده هستند چون قدیمی‌ترین آن‌ها هنوز زنده است). در نتیجه الگوریتم برون‌ـخط ما، حداقل یکی از این $z+۱$ نقطه را با شعاع $۴R$ می‌پوشاند. از طرفی فاصله‌ی نقطه‌ی فراموش شده با این نقطه حداکثر برابر $۴R$ است (چون هر دو در یک توپ با شعاع $۲R$ قرار دارند). در نتیجه همه‌ی نقاط فراموش شده با شعاع حداکثر $۸R$ و همه به جز حداکثر z تا از نقاط موجود در حافظه با شعاع حداکثر $۴R$ پوشیده می‌شوند و در کل همه‌ی نقطه‌ها به جز حداکثر z تا با شعاع $۸R$ پوشیده می‌شوند. □

قضیه ۲۲. *اعمال موازی سازی بر روی الگوریتم ۷، الگوریتمی با ضریب تقریب $۸+\epsilon$ ارائه می‌دهد.*

اثبات. از لم‌های قبلی و قضیه‌ی ۱۹ نتیجه می‌شود. □

قضیه ۲۳. *تعداد نقاط موجود در حافظه در تمام مراحل الگوریتم ۷ حداکثر $O(kz+z^۲)$ است.*

اثبات. نقاط موجود در حافظه شامل مراکز موجود در C به همراه نقاط شاهد آن‌ها و همچنین همه‌ی نقاط موجود در D می‌شوند. می‌دانیم که $|C|$ از $k+z+۱$ بیشتر نیست. در نتیجه حداکثر نقاط موجود در C برابر $(z+۱)(k+z+۱) = O(kz+z^۲)$ است (چون به ازای هر مرکز، حداکثر $z+۱$ نقطه‌ی شاهد

داریم). می‌دانیم که در D نقاط شاهد مرکزهای حذف شده قرار دارند اما می‌دانیم که تعداد مراکزی که نقاط شاهدشان در D قرار دارد از $k+z+۱$ بیشتر نیست. دلیل این موضوع آن است که می‌دانیم همه‌ی نقاط شاهد مرکز iام، از مرکز $i+k+z+۱$ام قدیمی‌تر هستند (چون در زمان ورود آن مرکز، مرکز iام باید از C خارج شده باشد و در نتیجه پس از آن هیچ نقطه‌ی شاهد جدیدی نمی‌پذیرد) از طرفی برای این که نقاط شاهد مرکز $i+k+z+۱$ام به D اضافه شوند، این مرکز باید از C خارج شود که به این معنی است که همه‌ی نقاطی که از آن قدیمی‌تر هستند، از D حذف می‌شوند. در نتیجه D شامل نقاط شاهد حداکثر $k+z+۱$ مرکز است و هر مرکز هم حداکثر $z+۱$ نقطه‌ی شاهد دارد و در نتیجه حافظه‌ی مورد نیاز توسط D از $O(kz+z^۲)$ است. پس حافظه‌ی کلی الگوریتم برابر

$$|C|+|D|=O(kz+z^۲)+O(kz+z^۲)=O(kz+z^۲)$$

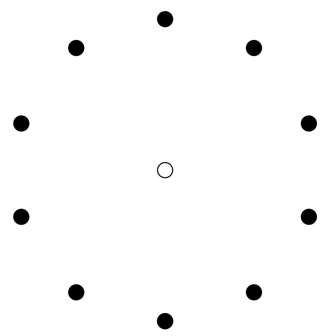
می‌باشد. □

۴ــ۵ــ۱ الگوریتم برون‌ــخط

در الگوریتم قبلی، از الگوریتمی برون‌ــخط با ضریب تقریب ۴ یاد کردیم. در این قسمت این الگوریتم را ارائه می‌دهیم. این الگوریتم، یک نسخه‌ی تغییر یافته از الگوریتم با ضریب تقریب ۳ ارائه شده در [۷] است. ممکن است که عجیب به نظر برسد که یک الگوریتم با ضریب تقریب ۳ را تغییر می‌دهیم تا به الگوریتمی با ضریب تقریب ۴ برسیم. دلیل این موضوع این است که الگوریتم [۷] فرض می‌کند که مرکز بهینه، یکی از نقاط داده شده است. در الگوریتم ۷ به دلیل این که بعضی نقاط را فراموش کرده‌ایم نمی‌توانیم چنین فرضی کنیم. همچنین می‌توان نشان داد که اگر شرط این که مرکز یکی از نقاط ورودی است را بر داریم، جواب بهینه می‌تواند تا دو برابر کاهش پیدا کند (شکل ۴ــ۵). در نتیجه اعمال بدیهی الگوریتم [۷] در الگوریتم ۷، ضریب تقریب ۱۰ را نتیجه می‌دهد.

الگوریتم برون‌ــخط ما مشابه [۷] است که در قسمت کارهای پیشین شرح داده شد. تنها تفاوت این است که به جای این که بر روی نقاط دایره‌هایی به شعاع‌های R و $۳R$ فرض کنیم، دایره‌هایی با شعاع‌های $۲R$ و $۴R$ فرض می‌کنیم و نام آن‌ها را به ترتیب G_i و E_i می‌گذاریم.

قضیه ۲۴. *اگر R از جواب بهینه (مراکز آن الزاماً از نقاط داده شده نیستند) بیشتر باشد، الگوریتم ۸ مراکزی را پیدا می‌کند که با شعاع حداکثر $۴R$ همه به جز حداکثر z تا از نقاط را پوشش دهند.*



شکل ۴ـ۵: فراموش کردن نقطه‌ی مرکز، جواب بهینه را دو برابر افزایش می‌دهد

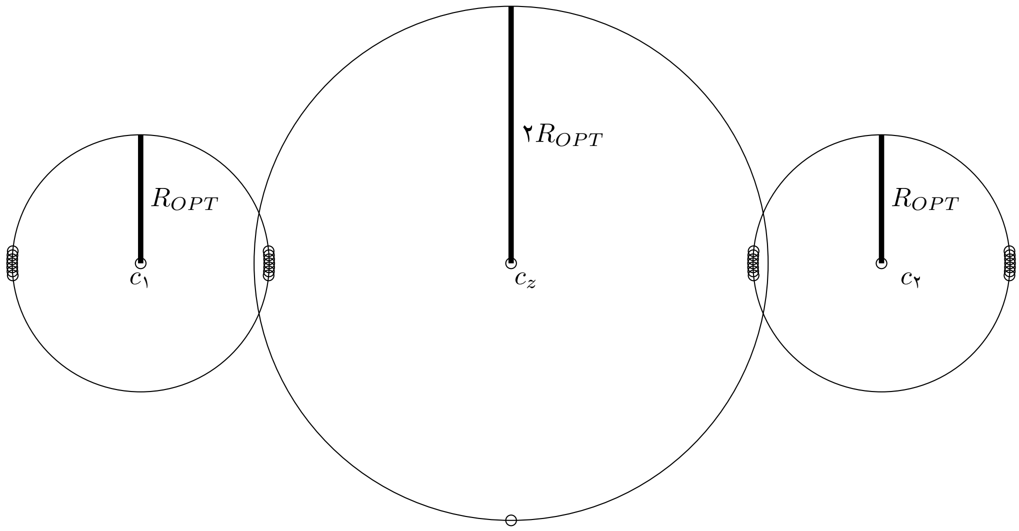
الگوریتم ۸ الگوریتم برون خط با ضریب تقریب ۴

همه‌ی توپ‌های G و E را تولید کن
$C = \emptyset$
به ازای j در $\{۱, ..., k\}$:
    G_c توپی با شعاع $۲R$ است که بیشترین تعداد نقطه را پوشش می‌دهد
    $C = C \cup \{c\}$
    همه‌ی نقاط پوشیده شده توسط E_c را از G_iها و E_iها حذف کن
اگر بیشتر از z نقطه‌ی پوشش داده نشده باقی مانده است:
    برگردان شکست
در غیر این صورت:
    برگردان C

اثبات. اثبات این قضیه مشابه قضیه‌ی ۱ است. تنها تفاوت این است که در این جا مراکز جواب بهینه الزاماً از نقاط داده شده نیستند. ولی چون شعاع توپ‌های G_i برابر $۲R$ است، می‌دانیم که همچنان نقاطی هستند که G_iاشان همه‌ی توپ بهینه‌ای که داخل آن هستند را پوشش می‌دهد و در نتیجه، اثبات قبلی در این الگوریتم هم جواب می‌دهد. □

قضیه ۲۵. *الگوریتم ۸ در زمان $O(kn^۲)$ و حافظه‌ی $O(n)$ قابل پیاده سازی است.*

اثبات. حلقه‌ی اصلی این الگوریتم k بار تکرار می‌شود و هر بار تکرار در $O(n^۲)$ قابل پیاده سازی است (کافی است که به ازای هر نقطه، تمام نقاطی که در شعاع $۲R$ آن هستند را با پیمودن همه‌ی نقاط به ازای آن نقطه به دست آوریم که برای هر نقطه $O(n)$ زمان می‌برد و در نتیجه اجرای آن برای تمام نقاط، به $O(n^۲)$ زمان نیاز دارد) در نتیجه در کل به $O(kn^۲)$ زمان نیاز دارد. حافظه‌ی مورد استفاده توسط این الگوریتم فقط لیست نقاط است که به $O(n)$ حافظه نیاز دارد. □



شکل ۴ـ۶: مثال تنگ برای الگوریتم ۸

شکل ۴ـ۶ مثالی تنگ برای الگوریتم ۸ را نشان می‌دهد. فرض کنید $k = 2$ و $z = 2$. در این صورت، همان طور که مشخص است، مراکز بهینه c_1 و c_2 هستند که همه‌ی نقاط به جز دو تا را با شعاع R_{OPT} پوشش می‌دهند. امّا الگوریتم ما در مرحله‌ی اوّل c_z را به عنوان مرکز انتخاب می‌کند چون تعداد نقاط موجود در شعاع $2R$ آن از بقیه‌ی نقاط بیشتر است. سپس همه‌ی نقاط که در شعاع $4R$ نقطه‌ی c_z هستند حذف می‌شوند که در این مثال هیچ نقطه‌ای باقی نمی‌ماند. در نتیجه مرکز دومی انتخاب نمی‌شود و الگوریتم، نقطه‌ی c_z را به عنوان مرکز بازمی‌گرداند که همان طور که در شکل مشخص است، شعاع آن $4R_{\text{OPT}}$ است. همچنین می‌توان با ایده‌ای مشابه، برای الگوریتم ۷ مثال تنگ ارائه داد (باید فرض کنیم شعاع بهینه برابر $R_{\min}$ است و سپس شکلی مشابه شکل بالا تولید کنیم با این تفاوت که شعاع‌ها دو برابر افزایش پیدا کرده‌اند).

اکنون می‌توانیم زمان الگوریتم ۷ را تحلیل کنیم.

قضیه ۲۶. *الگوریتم ۷ با زمان درج $O(kz+z^2)$ زمان جست‌وجوی $O(k(kz+z^2)^2)$ و حافظه‌ی $O(kz+z^2)$ قابل پیاده سازی است.*

اثبات. همان طور که در لم ۲۳ اثبات شد، تعداد نقاط موجود در حافظه $O(kz+z^2)$ است. از طرفی عمل درج را با یک بار پیمایش نقاط موجود در حافظه می‌توان انجام داد. در نتیجه زمان مورد نیاز برای درج $O(kz+z^2)$ است.

همان طور در که در قضیه‌ی ۲۵ گفته شد، زمان مورد نیاز برای الگوریتم برون‌ـ‌خط $O(kn^۲)$ است که در این جا $n = kz + z^۲$ در نتیجه زمان مورد نیاز برای پرس‌وجو برابر $k(kz+z^۲)^۲$ است. □

قضیه ۲۷. *با اعمال موازی سازی بر روی الگوریتم ۷، الگوریتمی با ضریب تقریب $\epsilon + ۸$ و زمان درج $O((kz+z^۲)\log_{۱+\epsilon}\Delta)$ و زمان پرس‌وجوی $O((k(kz+z^۲)^۲)\log_۲\log_{۱+\epsilon}\Delta)$ و حافظه‌ی $O((kz+z^۲)\log_{۱+\epsilon}\Delta)$ به دست می‌آوریم.*

اثبات. از لم ۱۹ و قضیه‌ی ۲۲ و قضیه‌ی ۲۶ نتیجه می‌شود. □

۴ـ۶ kـ‌مرکز در فضای متریک با بعد مضاعف ثابت

در این قسمت یک نسخه‌ی تغییر یافته از الگوریتم ۷ را ارائه می‌کنیم که در فضاهای متریک با بعد مضاعف ثابت ضریب تقریب $\epsilon + ۳$ دارد.

ایده‌ی کلی این الگوریتم این است که به جای حل مسئله‌ی kـ‌مرکز، مسئله‌ی k'ـ‌مرکز را حل می‌کند که مقدار k' طوری انتخاب می‌شود که شعاع هر کدام از k' مرکز انتخاب شده، حداکثر برابر $\epsilon R_{\rm OPT}$ باشد. سپس همانند الگوریتم ۷، مسئله‌ی برون‌ـ‌خط را بر روی نقاط موجود در حافظه حل می‌کند با این تفاوت که به جای استفاده از الگوریتم ما با ضریب تقریب ۴، از الگوریتم اصلی با ضریب تقریب ۳ استفاده می‌کند.

لم ۲۸. *اگر شعاع بهینه برای مسئله‌ی kـ‌مرکز با z نقطه‌ی پرت برابر $R_{\rm OPT}$ باشد، جواب بهینه برای مسئله‌ی $kD^{\lceil \frac{۳۲}{\epsilon} \rceil}$ـ‌مرکز با z نقطه‌ی پرت حداکثر برابر $\frac{\epsilon R_{\rm OPT}}{۱۶}$ است.*

اثبات. طبق قضیه‌ی ۹، می‌دانیم که هر توپ با شعاع $R_{\rm OPT}$ را می‌توان با $D^{\lceil \frac{۳۲}{\epsilon} \rceil}$ توپ با شعاع $\frac{\epsilon R_{\rm OPT}}{۳۲}$ پوشاند. در نتیجه با $kD^{\lceil \frac{۳۲}{\epsilon} \rceil}$ توپ با شعاع $\frac{\epsilon R_{\rm OPT}}{۳۲}$ می‌توان همه‌ی k توپ بهینه را پوشش داد. از طرفی هر نقطه‌ای داخل این دایره‌ها، می‌تواند تمام دایره را با شعاع $\frac{\epsilon R_{\rm OPT}}{۱۶}$ پوشش دهد. در نتیجه کافی است که به ازای هر کدام از این توپ‌ها، حداکثر یک نقطه داخل آن‌ها را انتخاب کنیم تا تمام مراکز بهینه با شعاع $\frac{\epsilon R_{\rm OPT}}{۱۶}$ پوشیده شوند. □

لم ۲۹. *اگر $R \geqslant R_{\rm OPT}$، الگوریتم موفق می‌شود.*

الگوریتم ۹ kـ مرکز با نقاط پرت در فضای متریک با بعد مضاعف ثابت

۱: $C = \emptyset$

۲: $D = \emptyset$

۳: **رویه** درج(p)

۴: اگر اگر مرکزی مثل c در C از پنجره خارج شده است:

۵: حذف(c)

۶: اگر نقطه‌ای مثل d در D از پنجره خارج شده است:

۷: $D = D \setminus \{d\}$

۸: $cp = \{c | c \in C, d(c,p) \leqslant \frac{\epsilon R}{8}\}$

۹: اگر $|cp| > 0$:

۱۰: $nc = cp$ جدید ترین نقطه در

۱۱: $nc.r = nc.r \cup \{p\}$

۱۲: اگر $|nc.r| > z + 1$:

۱۳: قدیمی‌ترین نقطه در $nc.r$ را حذف کن

۱۴: در غیر این صورت:

۱۵: $p.r = \emptyset$

۱۶: $C = C \cup \{p\}$

۱۷: اگر $|C| > kD^{\lceil \log_2 \frac{32}{\epsilon} \rceil} + z + 1$:

۱۸: $old = C$ قدیمی‌ترین مرکز در

۱۹: حذف(old)

۲۰: اگر $|C| > kD^{\lceil \log_2 \frac{32}{\epsilon} \rceil} + z$:

۲۱: $old = C$ قدیمی‌ترین مرکز در

۲۲: همه‌ی نقاط موجود در D که از old قدیمی‌تر هستند را حذف کن

۲۳: **رویه** حذف(c)

۲۴: $C = C \setminus \{c\}$

۲۵: $D = D \cup c.r$

۲۶: همه‌ی نقاط قدیمی‌تر از c را از D حذف کن

۲۷: **رویه** پرس‌وجو

۲۸: اگر $|C| > kD^{\lceil \log_2 \frac{32}{\epsilon} \rceil} + z$:

۲۹: برگردان شکست

۳۰: در غیر این صورت:

۳۱: نقاط موجود در حافظه را با شعاع $R(\frac{3\epsilon}{4} + 3)$ با استفاده از الگوریتم [۷] بپوشان

۳۲: اگر خط ۳۱ موفق شد:

۳۳: برگردان مراکز پیدا شده در خط ۳۱

۳۴: در غیر این صورت:

۳۵: برگردان شکست

اثبات. الگوریتم در دو صورت شکست می‌خورد: وقتی که $|C| > kD^{\lceil \frac{۳۲}{\epsilon} \rceil} + z$ و وقتی که الگوریتم برون_خط شکست می‌خورد. با استدلالی مشابه لم ۲۰ می‌دانیم که اتفاق اوّل غیر ممکن است. از طرفی می‌دانیم که هر نقطه‌ی فراموش شده، در شعاع $\frac{\epsilon R}{۴}$ یک نقطه در حافظه است. در نتیجه اگر جواب بهینه برای نقاط پنجره برابر R_{OPT} باشد، جواب بهینه برای نقاط موجود در حافظه نمی‌تواند از $R_{\mathrm{OPT}} + \frac{\epsilon R}{۴} < R + \frac{\epsilon R}{۴} = (۱ + \frac{\epsilon}{۴})R$ بیشتر باشد. در نتیجه هر الگوریتم با ضریب تقریب ۳ باید بتواند این نقاط را با شعاع حداکثر $(۳ + \frac{۳\epsilon}{۴})R = ۳(۱ + \frac{\epsilon}{۴})R$ بپوشاند و در نتیجه الگوریتم برون_خط هم شکست نمی‌خورد. □

لم ۳۰. *اگر الگوریتم موفق شود، جوابی بر می‌گردان که همه‌ي نقاط پنجره به جز حداکثر z تا را با شعاع $(۳ + \epsilon)R$ پوشش می‌دهد.*

اثبات. می‌دانیم که همه‌ی نقاط فراموش شده، در شعاع $\frac{\epsilon R}{۴}$ یک پوشیده شده توسط الگوریتم برون_خط هستند (با استدلالی مشابه استدلال الگوریتم برای فضای متریک). از طرفی شعاع الگوریتم برون_خط حداکثر برابر $(۳ + \frac{۳\epsilon}{۴})R$ است. در نتیجه همه‌ی نقاط فراموش شده هم با شعاع $(۳ + \frac{۳\epsilon}{۴})R + \frac{\epsilon R}{۴} = (۳ + \epsilon)R$ پوشش داده می‌شوند. □

قضیه ۳۱. *اعمال موازی سازی بر روی الگوریتم ۹، الگوریتمی با ضریب تقریب $(۳ + \epsilon_۱)(۱ + \epsilon_۲)$ و حافظه‌ی $O((kz(\frac{۳۲}{\epsilon_۱})^{\log_۲ D} + z^۲) \log_{۱+\epsilon_۲} \Delta)$ و زمان درج $O((kz(\frac{۳۲}{\epsilon_۱})^{\log_۲ D} + z^۲) \log_{۱+\epsilon_۲} \Delta)$ و زمان پرس‌وجوی $O(k(kz(\frac{۳۲}{\epsilon_۱})^{\log_۲ D} + z^۲)^۲ \log_{۱+\epsilon_۲} \Delta)$ می‌دهد.*

اثبات. از لم‌های ۳۰ و ۲۹ و قضیه‌ی ۱۹ و قضیه‌ی ۲۶ نتیجه می‌شود.

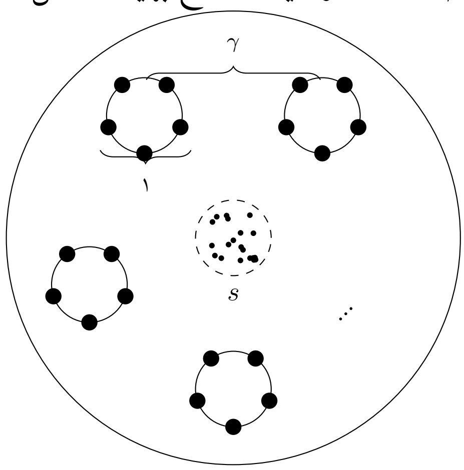
□

۴_۷ کران‌های پایین

در این قسمت کران پایینی برای مسئله‌ی k_مرکز با نقاط پرت در مدل پنجره‌ی لغزان ارائه می‌کنیم و نشان می‌دهیم که الگوریتم ما از نظر تعداد نقطه‌ی ذخیره شده در حافظه تقریباً بهینه است.

فرض کنید $n > (k + ۱)(z + ۱)$ باشد. $k + ۱$ نقطه را در نظر بگیرید به طوری که فاصله‌ی هر دوتا از این $k + ۱$ نقطه، برابر γ باشد به طوری که $\gamma > \beta^۲ + ۱$. $k + ۱$ مجموعه‌ی $z + ۱$

نقطه‌ی اوّل را با K و نقطه‌ی $k+1$ام را با s نشان می‌دهیم. حال $k(z+1)$ نقطه‌ی اوّل پنجره را طوری توزیع می‌کنیم که در شعاع $\frac{1}{2}$ هر یک از نقاط K دقیقاً $z+1$ نقطه وجود داشته باشد. بقیه‌ی $n-k(z+1) > (k+1)(z+1)-k(z+1) \geqslant z+1$ نقطه‌ی پنجره را در شعاع $\frac{1}{2}$ نقطه‌ی s قرار می‌دهیم (شکل ۴ـ۷). توجه کنید که در انتهای پنجره، $k+1$ مرکز داریم که در شعاع $\frac{1}{2}$ هر یک، حداقل $z+1$ نقطه موجود است. در نتیجه برای پوشش دادن این نقاط با حداکثر z نقطه‌ی پرت، حداقل یکی از مراکز ارائه شده باید دو نقطه از دو مرکز متفاوت را پوشش دهد. اما فاصله‌ی هر یک از نقاط تا نقطه‌ای از مرکزی دیگر حداقل $\gamma - 1 > \beta^2$ است. در نتیجه شعاع بهینه حداقل β^2 است.



شکل ۴ـ۷: فاصله‌ی بین مرکزها، γ برابر شعاع است

حال ادعا می‌کنیم که هر الگوریتم با ضریب تقریب β باید بتواند بین همه‌ی حالت‌های توزیع این $k(z+1)$ نقطه بین مراکز K تمایز قائل شود.

لم ۳۲. *هر الگوریتم برای مسئله‌ی kـ مرکز با نقاط پرت در پنجره‌ی لغزان که ضریب تقریبی بهتر از β ارائه می‌دهد، باید بتواند بین هر دو حالت ممکن توزیع $k(z+1)$ نقطه‌ی اوّل بین مراکز K تمایز قائل شود.*

اثبات. فرض کنید که دو پنجره‌ی متفاوت A و B با توزیع‌های متفاوت داریم که الگوریتم بین آن‌ها تمایز قائل نمی‌شود. به عبارت دیگر، ساختار حافظه در پایان پنجره‌های A و B یکسان است. iامین نقطه‌ی A و B را به ترتیب با A_i و B_i نشان می‌دهیم. برای هر نقطه‌ی p، شماره‌ی مرکزی که p در شعاع $\frac{1}{2}$ آن قرار دارد را با $c(p)$ نشان می‌دهیم. فرض کنید l اوّلین جایی باشد که به ازای آن $A_l \neq B_l$. می‌دانیم که چنین اندیسی وجود دارد چون در غیر این صورت توزیع A و B یکی می‌شد.

نقطه‌ی iام پنجره را با X_i نشان می‌دهیم. حال، نقطه‌های جدید را به ازای $i > n$ با این قوانین معرفی می‌کنیم:

۱. اگر $i < l$، $X_i = A_{i-n} = B_{i-n}$

۲. اگر $i = l$، X_i را به طور تصادفی و یکنواخت از $\{A_l, B_l\}$ انتخاب می‌کنیم.

دقت کنید که هنگامی که نقاط جدید را با قانون ۱ معرفی می‌کنیم، نقاط موجود در پنجره تغییری نمی‌کند چون نقطه‌ی جدید را دقیقاً همان جایی که نقطه‌ی حذف شده قرار داشت گذاشته‌ایم. در نتیجه جواب بهینه برای پنجره همچنان حداقل برابر $\beta^۲$ است. امّا هنگامی که از قانون ۲ استفاده می‌کنیم، نقاط پنجره فقط وقتی تغییر نمی‌کند که نقطه‌ی جدید را از همان مرکزی که نقاط پنجره‌ی اولیه از آن بودند انتخاب کنیم. این اتفاق با احتمال $\frac{۱}{۲}$ می‌افتد. در غیر این صورت، یکی از $k + ۱$ مرکز اکنون z نقطه دارد. در نتیجه می‌توان همه‌ی نقاط این مرکز را پرت فرض کرد و در بقیه‌ی نقاط در k مرکز با شعاع حداکثر ۱ قرار می‌گیرند و در نتیجه جواب بهینه برای پنجره برابر ۱ می‌شود. از آن‌جایی که X_{n+l} را به طور یکنواخت و تصادفی انتخاب کردیم، جواب بهینه با احتمال $\frac{۱}{۲}$ برابر ۱ و با احتمال $\frac{۱}{۲}$ برابر $\beta^۲$ است و الگوریتم ما هیچ راهی برای تمایز بین این دو حالت ندارد (چون ساختار حافظه پس از پنجره‌های A و B یکسان است). در نتیجه هر جوابی که الگوریتم ما ارائه کند، با احتمال $\frac{۱}{۲}$ با ضریب β با جواب واقعی فاصله دارد. □

لم ۳۳. *هر الگوریتم تقریبی برای مسئله‌ی kـ مرکز با نقاط پرت در مدل پنجره‌ی لغزان که با احتمال بیشتر از $\frac{۱}{۲}$ موفق شود، حداقل به $\Omega(kz \log k)$ بیت حافظه نیاز دارد.*

اثبات. مقدار β را برابر ضریب تقریب الگوریتم قرار می‌دهیم. طبق لم ۳۲، می‌دانیم که باید بتوانیم بین هر دو توزیع $k(z+۱)$ نقطه‌ی اوّل بین مراکز K تمایز قائل شویم. برای توزیع این نقاط، می‌توانیم همه‌ی زمان‌های ورود را به طور تصادفی بر بزنیم و مرکز اوّل $z + ۱$ زمان اوّل را می‌گیرد و مرکز دوم $z + ۱$ زمان دوم و به همین ترتیب مرکز iام، $z + ۱$ زمان iام را می‌گیرد. دقت کنید که برای یک مجموعه‌ی $z + ۱$ زمان ورود، ترتیب‌های مختلف این مجموعه با هم تفاوتی نمی‌کنند. در نتیجه به ازای هر مرکز باید جواب را تقسیم بر $(z + ۱)!$ کنیم. در نتیجه تعداد کل حالت‌ها می‌شود $\frac{(k(z+۱))!}{(z+۱)!^k}$. برای این که بین هر دو تایی از این توزیع‌ها بتوانیم تمایز قائل شویم، حداقل به $\log_۲ \frac{(k(z+۱))!}{(z+۱)!^k}$ بیت حافظه نیاز داریم که

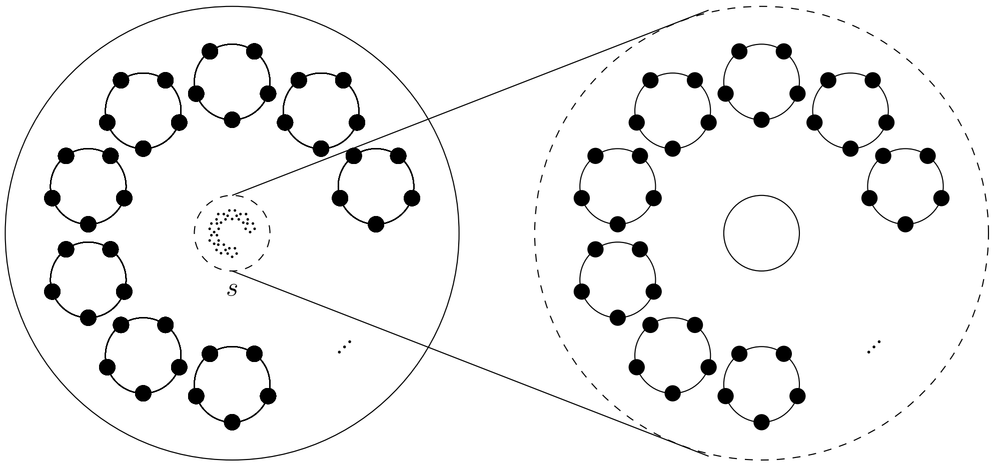
می‌شود:

$$\begin{aligned}
&\log\left(\frac{(k(z+1))!}{((z+1)!)^k}\right) = \\
&\log\left[\prod_{i=0}^{k-1}\left(\frac{((k-i)(z+1))((k-i)(z+1)-1)...((k-i)(z+1)-z)}{(z+1)!}\right)\right] \\
&= \log\left[\prod_{i=0}^{k-1}\left(\frac{\prod_{j=0}^{z}[((k-i)(z+1)-j)]}{(z+1)!}\right)\right] \\
&\geqslant \log\left[\prod_{i=0}^{k-2}\left(\frac{\prod_{j=0}^{z}[((k-i)(z+1)-(z+1))]}{(z+1)!}\right)\right] \\
&\geqslant \log\left[\prod_{i=0}^{k-2}\left(\frac{\prod_{j=0}^{z}[(k-i-1)(z+1)]}{(z+1)^{z+1}}\right)\right] \\
&= \log\left[\prod_{i=0}^{k-2}\left(\prod_{j=0}^{z}[(k-i-1)]\right)\right] \\
&= \sum_{i=0}^{k-2}\left(\sum_{j=0}^{z}[\log(k-i-1)]\right) = \sum_{i=0}^{k-2}(z+1)\log(k-i-1) \\
&= (z+1)\sum_{i=1}^{k-1}\log i = (z+1)\Omega(k\log k) = \Omega(kz\log k)
\end{aligned}$$

□

دقت کنید که ما محدودیتی در مورد این که چگونه نقاط داخل s را قرار دهیم نداریم. در نتیجه می‌توانیم به طور بازگشتی همین ساختار را در s تکرار کنیم. برای این کار کافی است که همه‌ی فاصله ها را با ضریب γ کاهش دهیم به طوری که فاصله‌ی بین مراکز برابر $\frac{\gamma}{\gamma} = 1$ و شعاع مراکز برابر $\frac{1}{2\gamma} = \frac{0/5}{\gamma}$ شود. این موضوع در شکل ۴ـ۸ نشان داده شده است. با استدلالی مشابه قبل، برای این ساختار لایه‌ی دوم هم به $\Omega(kz\log k)$ حافظه نیاز داریم. می‌توانیم این فرآیند را $\left\lfloor\frac{n-(z+1)}{k(z+1)}\right\rfloor$ بار تکرار کنیم و هر بار تکرار به $\Omega(kz\log k)$ حافظه نیاز دارد در نتیجه حافظه‌ی مورد نیاز کلی برابر $\Omega\left(\left\lfloor\frac{n-(z+1)}{k(z+1)}\right\rfloor(kz\log k)\right) = \Omega(n\log k)$ بیت می‌شود (با فرض این که $k(z+1) << N-(z+1)$). این نشان می‌دهد که در حالت کلی (اگر α را محدود نکنیم)، این مسئله با حافظه‌ی زیر خطی در مدل پنجره‌ی لغزان قابل حل نیست. دقت کنید که در ساختار ما، $\alpha = \gamma^{\Omega(\frac{n}{kz})}$ که با الگوریتم ۷ نیاز به $(kz+z^2)\log\alpha < kz\log\alpha' = kz\Omega(\frac{n}{kz}) = \Omega(n)$ حافظه دارد. قضیه‌ی بعدی نسخه‌ی قوی‌تر کران ما را با در نظر گرفتن α اثبات

می‌کند.



شکل ۴ــ۸: ساختار به طور بازگشتی در s تکرار می‌شود

قضیه ۳۴. *هر الگوریتم با ضریب تقریب β برای مسئله‌ی kــ مرکز با نقاط پرت در مدل پنجره‌ی لغزان که با احتمال بیشتر از $\frac{1}{2}$ بر روی جویباری که پخشش نقاط آن برابر α است موفق شود، حداقل به $\Omega(kz \log k \log_\beta \alpha)$ بیت حافظه نیاز دارد.*

اثبات. بر اساس لم ۳۳، می‌دانیم که برای هر لایه از نقطه‌ها، به $\Omega(kz \log k)$ بیت حافظه نیاز داریم. به ازای هر لایه، نسبت کم‌ترین و بیش‌ترین جواب بهینه γ برابر می‌شود در نتیجه می‌توانیم حداقل $\lfloor \log_\gamma \alpha \rfloor$ لایه از ساختار بازگشتی داشته باشیم بدون این که این نسبت از α بیشتر شود. حافظه‌ی کلی می‌شود:

$$\lfloor \log_\gamma \alpha \rfloor \Omega(kz \log k) = \Omega(kz \log k \log_{\beta^۲+۱} \alpha) = \Omega(kz \log k \log_\beta \alpha)$$

□

۴ــ۸ الگوریتم دوتقریبی

تا به این‌جای کار در همه‌ی الگوریتم‌های ارائه شده، تقریب ما بر روی شعاع بهینه بود. در این قسمت، الگوریتمی را مطرح خواهیم کرد که علاوه بر شعاع، بر روی تعداد نقاط پرت نیز تقریب دارد. به این معنی که ممکن است بیشتر از z نقطه را پوشش ندهد. ساختار کلی مشابه الگوریتم ۷ است با این تفاوت که به جای نگه داشتن $z + ۱$ نقطه‌ی شاهد هر مرکز، با استفاده از داده ساختار شمارش ساده‌ی [۱۶]

تعداد نقاط موجود در آن مرکز را تقریب می‌زنیم (از نسخه‌ای که در مقدمه گفته شد استفاده می‌کنیم در نتیجه تقریب ارائه شده از مقدار واقعی کم‌تر است). برای ارائه‌ی جواب، هر مرکز را به تعداد نقاط شاهد آن تکرار می‌کنیم. سپس سعی می‌کنیم با استفاده از الگوریتم ۸ همه‌ی نقاط به جز حداکثر z تا را با شعاع $۱۲R$ بپوشانیم. در صورت موفقیت، این مراکز را به عنوان جواب باز می‌گردانیم.

لم ۳۵. *اگر $R \geqslant R_{\mathrm{OPT}}$، الگوریتم ۱۰ موفق می‌شود.*

اثبات. با استدلالی مشابه لم ۲۰ می‌دانیم که اگر $R \geqslant R_{\mathrm{OPT}}$، آنگاه در انتهای پنجره داریم $|C| \leqslant k+z$. در نتیجه تنها در صورتی الگوریتم شکست می‌خورد که الگوریتم برون‑خط شکست بخورد. فرض کنید که به جای تقریب زدن تعداد نقاط هر مرکز، مقدار دقیق آن‌ها را داریم (این موضوع فقط باعث می‌شود که پوشش دادن نقاط سخت تر بشود چون تقریب داده ساختار ما همیشه از تعداد واقعی نقاط آن مرکز کم‌تر است). می‌دانیم که همه به جز حداکثر z تا از نقاط پنجره با k مرکز با شعاع R_{OPT} قابل پوشش هستند. ما هر نقطه را به مرکز آن نقطه منتقل کرده‌ایم، در نتیجه هر نقطه حداکثر به اندازه‌ی $۲R$ جابجا شده است. در نتیجه می‌دانیم همه‌ی نقاط جابجا شده را می‌توان با k مرکز با شعاع $R_{\mathrm{OPT}} + ۲R \leqslant ۳R$ پوشش داد و در نتیجه الگوریتم ما با ضریب تقریب ۴ می‌تواند همه‌ی نقاط موجود در حافظه را با شعاع حداکثر $۴ * ۳R = ۱۲R$ پوشش دهد. □

لم ۳۶. *اگر الگوریتم ۱۰ موفق شود، جوابی باز می‌گرداند که همه‌ی نقاط پنجره به جز حداکثر $(۱+\delta)z$ تا را با شعاع $۱۴R$ پوشش می‌دهد.*

اثبات. اگر موفق شویم، هیچ کدام از نقاط ایجاد شده در پنجره‌ی فعلی حذف نشده اند و در نتیجه تنها خطای ما مربوط به جابجایی نقاط و همچنین خطای ساختار داده‌ی شمردن نقاط است. می‌دانیم که همه‌ی نقاط به جز حداکثر z تا را توانسته‌ایم با شعاع $۱۲R$ پوشش دهیم. می‌دانیم تعداد نقاط واقعی داخل هر مرکز حداکثر $۱ + \delta$ برابر مقدار گزارش شده برای آن مرکز است. در نتیجه تعداد واقعی نقاط پوشش داده نشده، حداکثر برابر $(۱ + \delta)z$ است. از طرفی می‌دانیم که همه‌ی مراکز دیگر پوشش داده شده‌اند. از آنجایی که فاصله‌ی هر نقطه تا مرکز مربوط به آن نقطه حداکثر برابر $۲R$ است، می‌دانیم که همه‌ی نقاط به جز حداکثر $(۱ + \delta)z$ تا با شعاع $۱۲R + ۲R = ۱۴R$ پوشش داده شده اند. □

قضیه ۳۷. *اعمال موازی سازی بر روی الگوریتم ۱۰، الگوریتمی با ضریب تقریب $۱۴ + \epsilon$ بر روی شعاع ارائه می‌دهد.*

الگوریتم ۱۰ الگوریتم دوتقریبی برای kـ مرکز با نقاط پرت

۱: $C = \emptyset$
۲: $D = \emptyset$
۳: **رویه** درج(p)
۴:     اگر مرکزی مثل c در C از پنجره خارج می‌شود:
۵:         حذف(c)
۶:     اگر مرکزی مثل c در D نقطه‌ی شاهدی ندارد:
۷:         $D = D \setminus \{c\}$
۸:     $cp = \{c | c \in C, d(c,p) \leqslant ۲R\}$
۹:     اگر $|cp| > ۰$:
۱۰:         جدیدترین نقطه در $nc = cp$
۱۱:         در نقاط شاهد nc یک ۱ درج کن
۱۲:         به ازای c in $C \cup D \setminus \{nc\}$:
۱۳:             در نقاط شاهد nc یک ۰ درج کن
۱۴:     در غیر این صورت:
۱۵:         $p.r$ را برابر داده ساختار تغییر یافته‌ی [۱۶] قرار بده
۱۶:         $C = C \cup \{p\}$
۱۷:         اگر $|C| > k + z + ۱$:
۱۸:             قدیمی‌ترین مرکز در $old = C$
۱۹:             حذف(old)
۲۰:         اگر $|C| > k + z$:
۲۱:             قدیمی‌ترین مرکز در $old = C$
۲۲:             همه‌ی مراکز موجود در D را که جدیدترین نقطه‌ی شاهدشان از old قدیمی‌تر است حذف کن
۲۳: **رویه** حذف(c)
۲۴:     $C = C \setminus \{c\}$
۲۵:     $D = D \cup \{c\}$
۲۶:     همه‌ی مراکز موجود در D را که جدیدترین نقطه‌ی شاهدشان از c قدیمی‌تر است حذف کن
۲۷: **رویه** پرس‌وجو
۲۸:     اگر $|C| > k + z$:
۲۹:         برگردان شکست
۳۰:     در غیر این صورت:
۳۱:         هر مرکز مثل c موجود در $C \cup D$ را به ازای هر نقطه‌ی شاهد که دارد تکرار کن سپس این نقاط را با الگوریتم برون خط با شعاع $۱۲R$ پوشش بده
۳۲:         اگر خط ۳۱ موفق شد:
۳۳:             برگردان مراکز پیدا شده در خط ۳۱
۳۴:         در غیر این صورت:
۳۵:             برگردان شکست

اثبات. از لم‌های قبلی و قضیه‌ی ۱۸ نتیجه می‌شود. □

لم ۳۸. *تعداد مراکز موجود در $C \cup D$ از $O(k+z)$ است.*

اثبات. مشابه قضیه‌ی ۲۳ اثبات می‌شود. □

قضیه ۳۹. *فرض کنید هر نقطه را می‌توان با s بیت حافظه نگه داری کرد. آنگاه الگوریتم ۱۰ با زمان درج سرشکن $O(k+z)$ و زمان جست و جوی $O(k(k+z)^{۲})$ و حافظه‌ی $O((k+z)(s+\frac{1}{\delta}\log^{۲} n))$ قابل پیاده سازی است.*

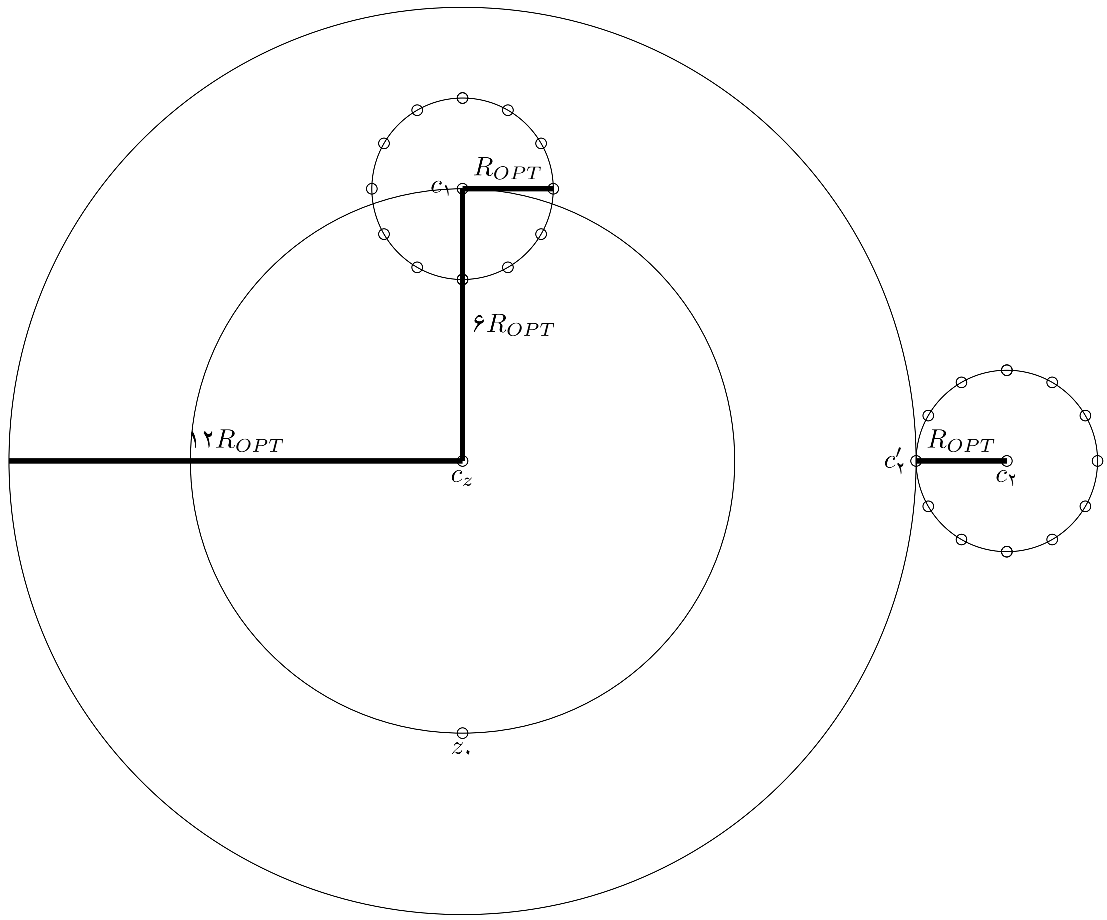
اثبات. همان طور که در [۱۶] نشان داده شده است، درج در داده ساختار شمارش را می‌توان در زمان سرشکن $O(۱)$ انجام داد. در الگوریتم ۱۰ درج با یکبار عبور از روی مراکز موجود در حافظه و درج در آن‌ها قابل پیاده سازی است و از آن‌جایی که $O(k+z)$ مرکز در حافظه داریم، عمل درج در زمان سرشکن $O(k+z)$ قابل انجام است.

با وجود این که برای اجرای الگوریتم برون_خط نقاط را تکرار کرده‌ایم، اگر فرض کنیم جمع کردن اعداد در $O(۱)$ قابل انجام است این الگوریتم در $O(k(k+z)^{۲})$ قابل پیاده سازی است چون نیاز به نمایش صریح همه‌ی نقاط نداریم و کافی است بدانیم که در هر مکان چند نقطه وجود داشته است.

همچنین نشان دادیم که تعداد مراکز موجود در $C \cup D$ از $O(k+z)$ است. به ازای هر کدام از این مراکز یک نقطه ذخیره کرده‌ایم که $O(s)$ حافظه نیاز دارد و یک داده ساختار شمارش که $O(\frac{1}{\delta}\log^{۲} n)$ حافظه نیاز دارد و در نتیجه حافظه‌ی کلی مورد نیاز الگوریتم، برابر $O((k+z)(s+\frac{1}{\delta}\log^{۲} n))$ است. □

قضیه ۴۰. *با اعمال موازی سازی بر روی الگوریتم ۱۰، الگوریتمی با ضریب تقریب $۱۴+\epsilon$ بر روی شعاع و ضریب تقریب $۱+\delta$ بر روی نقاط پرت و زمان درج سرشکن $O((k+z)\log_{۱+\epsilon}\Delta)$ و زمان پرس و جوی $O(k(k+z)^{۲}\log_{۲}\log_{۱+\epsilon}\Delta)$ و حافظه‌ی $O((k+z)(s+\frac{1}{\delta}\log^{۲} n)\log_{۱+\epsilon}\Delta)$ به دست می‌آوریم.*

شکل ۴_۹ یک مثال تنگ برای الگوریتم ۱۰ نشان می‌دهد. مراکز $c_۱$ و $c_۲$ مراکز بهینه‌ی ما هستند. فاصله‌ی همه‌ی نقطه‌های موجود در شعاع R_{OPT} مرکز $c_۱$ با یکدیگر برابر $۲R_{\text{OPT}}$ است. در نتیجه با هر تقریب کم‌تر از R_{OPT} تعداد مراکز از $k+z$ بیشتر می‌شود و در نتیجه الگوریتم شکست می‌خورد. فرض کنید تقریب برابر R_{OPT} است و $C=\{c_۱, c'_۲, c_z, z_۰\}$ و $D=\emptyset$. در این حالت، تعداد نقاط موجود در



شکل ۴ـ۹: مثال تنگ برای الگوریتم ۱۰

شعاع $6R_{\mathrm{OPT}}$ برای c_z از سایر نقاط C بیشتر است و در نتیجه c_z به عنوان یک مرکز انتخاب می‌شود. همه‌ی مرکزهای C در شعاع $12R_{\mathrm{OPT}}$ از c_z هستند و در نتیجه c_z به عنوان تنها مرکز انتخاب می‌شود. همان طور که در شکل مشخص است، فاصله‌ی c_z با نقاط مرکز c_2 برابر $14R_{\mathrm{OPT}}$ است.

۴ـ۹ قطر

در این قسمت بهبودی برای الگوریتم قطر چن و سجاد [۳۰] ارائه می‌کنیم. پیچیدگی زمانی و فضایی این الگوریتم نسبت به الگوریتم [۹] بیشتر است ولی این مزیت را دارد که نیازی به دانستن α ندارد.

نقاط جدید را به لیست B اضافه می‌کنیم. هنگامی که اندازه‌ی B به $\sqrt{n}$ رسید، آن را خلاصه سازی می‌کنیم. به این صورت که جدیدترین نقطه‌ی B را به عنوان مرکز در نظر می‌گیریم و به ازای هر بازه با ضریب $1+\epsilon$، آخرین نقطه‌ی موجود در بازه را نگه می‌داریم (اگر جدیدترین نقطه در یک بازه‌ی بزرگ‌تر از جدیدترین نقطه در بازه‌ی کوچک‌تر، جدیدتر بود، نیازی به نگه داری آخرین نقطه‌ی بازه‌ی کوچک‌تر نیست). برای جست‌وجو، دورترین فاصله‌ی یک مرکز خلاصه شده تا یک نقطه‌ی زنده منسوب به آن را به دست می‌آوریم. همچنین شعاع مراکز به همراه نقاط B را به دست می‌آوریم و از بین این دو، مقدار بزرگ‌تر را بر می‌گردانیم.

قضیه ۴۱. *الگوریتم ۱۱، جوابی با ضریب تقریب* $\max(3(1+\epsilon), 3\gamma)$ *بر می‌گرداند که* γ *ضریب تقریب الگوریتم استفاده شده در خط ۱۹ است.*

اثبات. فرض کنید نقاط دو سر قطر بهینه برای پنجره‌ی فعلی a و b باشند. فرض کنید c_a و c_b مراکز مربوط به این نقاط باشند (اگر هنوز خلاصه سازی نشده‌اند، خود نقاط مرکز هستند). طبق نامساوی مثلثی داریم:

$$d(a, c_a) + d(c_a, c_b) + d(c_b, b) \geqslant d(a, b)$$

از طرفی می‌دانیم که اگر نقطه‌ی a فراموش شده باشد، نقطه‌ای مانند p_a وجود دارد به طوری که $\frac{1}{\epsilon} d(a, c_a) \leqslant d(p_a, c_a) \leqslant (1+\epsilon) d(a, c_a)$ در نتیجه داریم:

$$(1+\epsilon) d(p_a, c_a) + d(c_a, c_b) + (1+\epsilon) d(c_b, p_b) \geqslant d(a, b)$$

الگوریتم ۱۱ قطر در پنجره‌ی لغزان

$B = \emptyset$

$S = \emptyset$

رویه درج(p)

  همه‌ی نقاط خارج شده از پنجره را از S حذف کن

  $B = B \cup \{p\}$

  اگر $|B| \geqslant \sqrt{n}$:

    $S = S \cup$ خلاصه(B)

    $B = \emptyset$

رویه خلاصه(P)

  $c =$ جدید ترین نقطه در P

  $d = \min_{t \in P \setminus \{c\}} d(t, c)$

  $S = \emptyset$

  به ازای i به طوری که نقطه‌ی مثل p وجود دارد که $d(1+\epsilon)^i \leqslant d(p, c) \leqslant d(1+\epsilon)^{i+1}$ و هیچ نقطه‌ی دیگری که فاصله‌اش با c بیشتر از $d(1+\epsilon)^{i+1}$ باشد وجود نداشته باشد:

    جدید ترین نقطه در این بازه را به S اضافه کن

  برگردان S

رویه پرس‌وجو

  $d_1 =$ بیشترین فاصله بین یک مرکز در C و یکی از نقاط شاهد آن

  $P = S$ اجتماع B و مراکز

  $d_2 = P$ نتیجه‌ی الگوریتم (تقریبی) قطر بر روی نقاط

  برگردان $\max(d_1, d_2)$

در نتیجه:

$$\max\{(۱+\epsilon)d(p_a,c_a), d(c_a,c_b), (۱+\epsilon)d(c_b,p_b)\} \geqslant \frac{d(a,b)}{۳}$$

اگر این مقدار بیشینه یکی از $(۱+\epsilon)d(p_a,c_a)$ یا $(۱+\epsilon)d(p_b,c_b)$ باشد، فاصله‌ی c_a یا c_b از دورترین نقطه به آن حداقل برابر $\frac{d(a,b)}{۳(۱+\epsilon)}$ خواهد بود و در غیر این صورت، شعاع مراکز برابر $\frac{d(a,b)}{۳}$ است و در نتیجه الگوریتم تقریبی ما جوابی ضریب تقریب ۳γ تولید خواهد کرد. □

قضیه ۴۲. *زمان مورد نیاز الگوریتم خط ۱۹ برای m نقطه را با $T_A(m)$ نشان می‌دهیم. حافظه‌ی مورد نیاز الگوریتم ۱۱ برابر $O(\sqrt{n}\log_{۱+\epsilon}\Delta)$ و زمان درج سرشکن آن برابر $O(۱)$ و زمان جست‌وجوی آن برابر $O(\sqrt{n}+T_A(\sqrt{n}))$ است.*

اثبات. هر ساختار خلاصه به $O(\log_{۱+\epsilon}\Delta)$ حافظه نیاز دارد و حداکثر $\sqrt{n}$ ساختار خلاصه داریم. همچنین اندازه‌ی B حداکثر برابر $\sqrt{n}$ است. در نتیجه حافظه‌ی کلی مورد نیاز برابر است با: $O(\sqrt{n}+\sqrt{n}\log_{۱+\epsilon}\Delta)=O(\sqrt{n}\log_{۱+\epsilon}\Delta)$

درج در صورتی که نیاز به خلاصه سازی نباشد در $O(۱)$ قابل انجام است و در غیر این صورت می‌توان با داده ساختار چن و سجاد [۳۰]، آن را در زمان سرشکن $O(۱)$ پیاده سازی کرد. برای جست‌وجو کافی است به ازای هر مرکز خلاصه شده دور ترین نقطه‌ی آن را پیدا کنیم که در کل برای تمام مراکز در $O(\sqrt{n})$ قابل پیاده سازی است. الگوریتم خط ۱۹ هم به $O(T_A(\sqrt{n}))$ زمان نیاز دارد. □

برای فضاهای متریک، می‌توان در خط ۱۹ از الگوریتم دقیق $O(n^۲)$ استفاده کرد. در نتیجه ضریب تقریب الگوریتم برابر $۳(۱+\epsilon)$ و زمان جست‌وجو برابر $O(n)$ می‌شود. در فضای اقلیدسی با ابعاد بالا، می‌توان از الگوریتم با ضریب تقریب $\sqrt{۳}$ استفاده کرد. به این ترتیب ضریب تقریب الگوریتم ما برابر $۳\sqrt{۳}\approx ۵/۲$ و زمان جست‌وجو برابر $O(d\sqrt{n})$ می‌شود. در فضای اقلیدسی دو بعدی و سه بعدی می‌توان قطر بهینه را در $O(n\log n)$ محاسبه کرد در نتیجه می‌توانیم به ضریب تقریب $۳(۱+\epsilon)$ با زمان جست‌وجوی $O(\sqrt{n}\log n)$ دست بیابیم.

فصل ۵

نتیجه گیری

۵ـ۱ نتیجه گیری

در این پایان نامه ابتدا مسئله‌ی ۱ـ مرکز با نقاط پرت در مدل پنجره‌ی لغزان را بررسی کردیم و برای حالتی که در فضای اقلیدسی d بعدی هستیم، الگوریتمی با ضریب تقریب ۲ ارائه دادیم که $O(d^2 z)$ نقطه را در حافظه نگه می‌دارد و برای حالتی که در فضای متریک هستیم، الگوریتمی با ضریب تقریب $3+\epsilon$ و حافظه‌ی $O(z+\log_{1+\epsilon}\alpha)$ نقطه ارائه دادیم همچنین نشان دادیم که اگر بعد مضاعف فضای متریک ما، عدد ثابتی مصل D باشد، می‌توان با ذخیره‌ی $O(z^2 D^{\log_2 \frac{1}{\epsilon}})$ نقطه، به ضریب تقریب $2+\epsilon$ دست یافت.

سپس برروی مسئله‌ی kـ مرکز برای k دلخواه کار کردیم و اوّلین الگوریتم را برای این مسئله با نقاط پرت در مدل پنجره‌ی لغزان ارائه دادیم. ضریب تقریب این الگوریتم برابر $8+\epsilon$ و تعداد نقاط ذخیره شده در حافظه حداکثر $O((kz+z^2)\log_{1+\epsilon}\alpha)$ است. همچنین نشان دادیم که وقتی که بعد مضاعف فضا ثابت است، می‌توان ضریب تقریب را به $3+\epsilon$ کاهش داد.

در بخش بعد، نشان دادیم که هر الگوریتم با ضریب تقریب β به حداقل $\Omega(kz\log k\log_\beta\alpha)$ بیت حافظه نیاز دارد که نشان می‌دهد که الگوریتم قسمت قبل تقریباً بهینه است.

تا به این جای کار، تنها تقریب الگوریتم‌های ما بر روی شعاع بهینه بود. در قسمت بعد، نشان دادیم که اگر علاوه بر شعاع بهینه، بتوانیم بر روی تعداد نقاط پرت نیز تقریب داشته باشیم، می‌توان به الگوریتمی با حافظه‌ی کم‌تر از کران پایین بخش قبل دست یافت. الگوریتم ما جوابی با ضریب تقریب

$14+\epsilon$ را با ذخیره‌ی تنها $O(\frac{1}{\delta\epsilon}(k+z)\log n \log \Delta)$ نقطه به دست می‌آورد و حداکثر $(1+\delta)z$ نقطه را پوشش نمی‌دهد.

در قسمت آخر نیز بهبود ساده‌ای برای الگوریتم قطر چن و سجاد [۳۰] مطرح کردیم و نشان دادیم که چگونه می‌توان ضریب تقریب آن را از $6+\epsilon$ به $3+\epsilon$ کاهش داد. مزیت اصلی این الگوریتم نسبت به الگوریتم [۹] این است که پیاده سازی آن نیازی با دانستن α ندارد.

۵ـ۲ کارهای آتی

در این پایان نامه، الگوریتمی با ضریب تقریب $8+\epsilon$ برای مسئله‌ی kـ مرکز با نقاط پرت ارائه دادیم. برای این مسئله قبلاً کار نشده بود ولی کوهن اداد و سایرین [۹] برای حالت بدون نقاط پرت الگوریتمی با ضریب تقریب $6+\epsilon$ ارائه داده بودند و همچنین نشان داده بودند که هر الگوریتم با ضریب تقریب بهتر از ۴ باید حداقل $\sqrt[3]{n}$ نقطه را ذخیره کند. در نتیجه یک شکاف بین این دو کران وجود دارد که بستن آن همچنان باز است. همچنین یک مسئله‌ی باز دیگر، انتقال این کران به حالتی که نقاط پرت داریم است.

همان طور که دیدیم، با ارائه‌ی تقریب بر روی تعداد نقاط پرت، با حافظه‌ی کم‌تر از کران پایین ارائه شده، توانستیم مسئله‌ی kـ مرکز با نقاط پرت را در مدل پنجره‌ی لغزان حل کنیم. گسترش کران پایین ما به حالتی که اجازه داریم بر روی z هم تقریب داشته باشیم، یک مسئله‌ی باز جالب دیگر است. همچنین متذکر می‌شویم که علاوه بر شعاع (R) و نقاط پرت (z) می‌توان بر روی تعداد مرکزها (k) و اندازه‌ی پنجره (n) نیز تقریب داشت. در نتیجه ارائه‌ی یک الگوریتم چهارتقریبی برای این مسئله، همچنان باز است. همچنین گسترش کران پایین به این حالت نیز مسئله‌ی باز جالبی است.

مراجع

[1] J. Paek and J. Ko. k-means clustering-based data compression scheme for wireless imaging sensor networks. *IEEE Systems Journal*, 11:2652–2662, 2017.

[2] N. Dhanachandra, K. Manglem, and Y. J. Chanu. Image segmentation using k-means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764–771, 2015.

[3] E. Diday, G. Govaert, Y. Lechevallier, and J. Sidi. Clustering in pattern recognition. In *Proceedings of the 5th International Conference on Pattern Recognition*, pages 19–58, 1980.

[4] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

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

[6] D. S. Hochbaum and D. B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33:533–550, 1986.

[7] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 642–651, 2001.

[8] R. Matthew McCutchen and S. Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In *Proceedings of the 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008 on Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques*, pages 165–178, 2008.

[9] V. Cohen-Addad, C. Schwiegelshohn, and C. Sohler. Diameter and k-center in sliding windows. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming*, pages 19:1–19:12, 2016.

[10] J. Heinonen. *Lectures on analysis on metric spaces*. Springer Science & Business Media, 2001.

[11] L. Danzer, B. Grünbaum, and V. Klee. *Helly's theorem and its relatives*. Proceedings of symposia in pure mathematics: Convexity. American Mathematical Society, 1963.

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

[13] C. C. Aggarwal. *Data streams: models and algorithms*. Springer, 2007.

[14] P. K. Agarwal, J. Matoušek, and S. Suri. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. *Computational Geometry*, 1:189–201, 1992.

[15] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: one-pass summaries for approximate aggregate queries. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 79–88, 2001.

[16] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows: (extended abstract). In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 635–644, 2002.

[17] T. Feder and D. Greene. Optimal algorithms for approximate clustering. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 434–444, 1988.

[18] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 626–635, 1997.

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

[20] H. Zarrabi-Zadeh. Core-preserving algorithms. In *Proceedings of the 20th Canadian Conference on Computational Geometry*, pages 159–162, 2008.

[21] S.-S. Kim and H.-K. Ahn. An improved data stream algorithm for clustering. *Computational Geometry*, 48:635–645, 2015.

[22] B. Hatami and H. Zarrabi-Zadeh. A streaming algorithm for 2-center with outliers in high dimensions. *Computational Geometry*, 60:26–36, 2017.

[23] P. K. Agarwal and R. Sharathkumar. Streaming algorithms for extent problems in high dimensions. *Algorithmica*, 72:83–98, 2010.

[24] T. M. Chan and V. Pathak. Streaming and dynamic algorithms for minimum enclosing balls in high dimensions. In *Proceedings of the 12th International Conference on Algorithms and Data Structures*, pages 195–206, 2011.

[25] H. Zarrabi-Zadeh and A. Mukhopadhyay. Streaming 1-center with outliers in high dimensions. In *Proceedings of the 21st Canadian Conference on Computational Geometry*, pages 83–86, 2009.

[26] T. M. Chan. An optimal randomized algorithm for maximum tukey depth. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 430–436, 2004.

[27] K. L. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S.-H. Teng. Approximating center points with iterated radon points. In *Proceedings of the 9th Annual Symposium on Computational Geometry*, pages 91–98, 1993.

[28] M. Ceccarello, A. Pietracaprina, and G. Pucci. Improved MapReduce and streaming algorithms for k-center clustering (with outliers). 2018.

[29] J.-L. Verger-Gaugry. Covering a ball with smaller equal balls in $\mathbb{R}^n$. *Discrete & Computational Geometry*, 33:143–155, 2005.

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

[31] F. P. Preparata and M. I. Shamos. *Computational geometry: an introduction.* 1985.

[32] R. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

[33] K. L. Clarkson. Applications of random sampling in computational geometry, II. In *Proceedings of the 4th Annual Symposium on Computational Geometry*, pages 1–11, 1988.

[34] E. A. Ramos. Deterministic algorithms for 3-D diameter and some 2-D lower envelopes. In *Proceedings of the 16th Annual Symposium on Computational Geometry*, pages 290–299, 2000.

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

[36] T. M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. In *Proceedings of the 16th Annual Symposium on Computational Geometry*, pages 300–309, 2000.

[37] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7:448–461, 1973.

[38] S. Finch. *Mathematical constants.* Encyclopedia of mathematics and its applications. 2003.

واژه‌نامه

ا

الگوریتم افزایشی ..... incremental algorithm

الگوریتم دوتقریبی . bi-criterial approximation
algorithm

ب

بعد مضاعف .... doubling dimension

پ

پخشش .... spread

پنجره‌ی لغزان.............. sliding window

پیر شدن .... aging

پیشگوی پیداکننده‌ی مرکز center finding oracle

ت

تابع هزینه‌ی یکنوا.... monotone cost function

تحلیل تصویر .... image analysis

تشخیص الگو.......... pattern recognition

تصادفی..................... randomized

توپ باز........................ open ball

توپ بسته .... closed ball

ج

جویبار داده.................. data stream

چ

چندگذری .... multipass

خ

خروجی .... output

خوشه بندی .... clustering

خوشه بندی دوتایی .... pairwise clustering

خوشه بندی مرکزی .... center clustering

د

دسترسی تصادفی .... random access

دستگاه ذخیره سازی خارجی .... external storage device

دیسک سخت .... hard disk

ش

شبکه .... network

ص

صندوق پول .... cash register

ف

فشرده سازی .... compression

فضای اقلیدسی .... euclidean space

فضای متریک .... metric space

فضای مضاعف .... doubling space

ق

قطر .... diameter

قطعی .... deterministic

م

مثال تنگ .... tight example

مجموعه‌ی هسته .... core-set

مرکز .... center

مسیریاب .... router

موازی سازی .... parallelization

میانگین .... mean

میانه .... median

ن

نامساوی مثلثی .... triangle inequality

نرم .... norm

نقاط پرت .... outliers

نقطه‌ی میانی .... centerpoint

نمایی .... exponential

و

ورودی .... input

ی

یادگیری بدون نظارت .. unsupervised learning

Abstract

With the emergence of massive datasets, storing all of the data in memory is not feasible for many problems. This fact motivated the introduction of new data processing models such as the streaming model. In this model, data points arrive one by one and the available memory is too small to store all of the data points. For many problems, more recent data points are more important than the old ones. The sliding window model captures this fact by trying to find the solution for the n most recent data points using only $o(n)$ memory. The k-center problem is an important optimization problem in which given a graph, we are interested in labeling k vertices of the graph as centers such that the maximum distance of all vertices to closest center is minimized. One of the major deterrents of more widespread use of the k-center problem in practice is its sensitivity to outliers to the point that even a small number of outlier points can increase the optimal radius unboundedly. This lead to the introduction of k center problem with outliers where we are able to ignore some of the points by labeling them as outliers. In this thesis, we try to develop approximation algorithms for the k-center problem with outliers in the sliding window model.

First, we study the case where $k = 1$ and we develop a 2-approximation algorithm for the Euclidean space and a $(3+\epsilon)$-approximation algortihm for general metric spaces. We also show that if the doubling dimension of the metric space is a known constant, the approximation ratio can be reduced to $2 + \epsilon$. Next, we present an $(8 + \epsilon)$-approximation algorithm for all k and we show that it is possible to reduce the approximation factor to $3 + \epsilon$ in metric spaces with constant doubling dimension. We also prove a lower bound showing that our algorithm is almost space-optimal. Finally, we show that if we are allowed to approximate the number of outliers in addition to radius, it is possible to beat our lower bound, by developing a bi-criterial approximation algorithm with approximation factor $1 + \delta$ on the number of outliers and $14 + \epsilon$ on the radius.

Keywords: Approximation algorithms, Metric space, Computational geometry, k-Center, Sliding window, Outliers

Sharif University of Technology

Department of Computer Engineering

M.Sc. Thesis

Approximating k-Center with Outliers in the Sliding Window Model

By:

Ali Mostafavi

Supervisor:

Dr. Hamid Zarrabi-Zadeh

August 2018