



دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

پایان‌نامه‌ی کارشناسی ارشد
گرایش مهندسی نرم‌افزار

عنوان:

الگوریتم‌های تقریبی برای خوشه‌بندی نقاط در مدل جویبار داده

نگارش:

بهنام حاتمی ورزنه

استاد راهنما:

حمید ضرابی‌زاده

شهریور ۱۳۹۴



به نام خدا
دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

پایان‌نامه‌ی کارشناسی ارشد

عنوان: الگوریتم‌های تقریبی برای خوشه‌بندی نقاط در مدل جویبار داده
نگارش: بهنام حاتمی ورزنه

کمیته‌ی ممتحنین

استاد راهنما: حمید ضرابی‌زاده
امضاء:

استاد مشاور: حمید بیگی
امضاء:

استاد مدعو: استاد ممتحن
امضاء:

تاریخ:

چکیده

مسئله k - مرکز به عنوان مسئله‌ای شناخته شده در علوم کامپیوتر مطرح است و در حوزه‌های مختلفی مورد استفاده قرار می‌گیرد. هدف مسئله k - مرکز پیدا کردن کم‌ترین شعاعی است که می‌توان مجموعه‌ی داده شده از نقاط را با k کره با آن شعاع پوشاند. این مسئله در حالت کلی n - پیوسته است. تمرکز اصلی این پایان‌نامه بر روی مسئله k - مرکز در حالت جویبار داده با داده‌ی پرت در ابعاد بالا است. به علت افزایش روز افزون حجم داده‌ها، گونه‌ی جویبار داده‌ی مسئله برای پاسخ‌گویی به حجم وسیع داده‌ها مورد توجه قرار می‌گیرد. از طرفی دیگر، وجود خطا در بین داده‌های تجربی، در نظر گرفتن داده‌های پرت را پر اهمیت می‌کند.

در این پایان‌نامه، دو مسئله 1 - مرکز و 2 - مرکز در فضای اقلیدسی با داده‌ی پرت در مدل جویبار داده مورد بررسی قرار می‌گیرد. برای مسئله 1 - مرکز با داده‌ی پرت، در حالتی که تعداد داده‌های پرت ثابت باشد، الگوریتمی با ضریب تقریب $1/7$ ارائه می‌دهیم که الگوریتم قبلی با ضریب تقریب $1/73$ را بهبود می‌بخشد. برای مسئله 2 - مرکز با داده‌ی پرت، الگوریتمی با ضریب تقریب $1/8 + \epsilon$ ارائه می‌دهیم که نسبت به الگوریتم قبلی با ضریب تقریب $4 + \epsilon$ ، بهبود قابل توجهی محسوب می‌شود. حافظه‌ی مصرفی و زمان به‌روزرسانی هر دو الگوریتم از مرتبه‌ی چندجمله‌ای نسبت به d ، n و $1/\epsilon$ است که مستقل از طول جویبار داده است.

کلیدواژه‌ها: خوشه‌بندی، k - مرکز، جویبار داده، الگوریتم تقریبی

فهرست مطالب

۱۰	۱ مقدمه
۱۲	۱-۱ تعریف مسئله
۱۴	۲-۱ اهمیت موضوع
۱۵	۳-۱ ادبیات موضوع
۱۶	۴-۱ اهداف تحقیق
۱۷	۵-۱ ساختار پایان نامه
۱۸	۲ مفاهیم اولیه
۱۸	۱-۲ مسائل ان پی- سخت
۲۰	۲-۲ مسائل خوشه بندی
۲۳	۳-۲ الگوریتم های تقریبی
۲۴	۲-۳-۱ میزان تقریب پذیری مسائل
۲۵	۴-۲ الگوریتم های جویبار داده
۲۶	۲-۴-۱ گونه های مطرح
۲۷	۲-۴-۲ تحلیل الگوریتم های جویبار داده
۲۷	۲-۴-۳ مجموعه هسته

۳۰	۳ کارهای پیشین
۳۰	۳-۱ k -مرکز در حالت ایستا
۳۴	۳-۲ k -مرکز در حالت جویبار داده
۴۲	۳-۳ k -مرکز با داده‌های پرت
۴۸	۴ نتایج جدید
۴۹	۴-۱ نمادگذاری‌ها و تعاریف اولیه
۵۱	۴-۲ مسئله‌ی ۱-مرکز در حالت جویبار داده
۵۱	۴-۲-۱ مسئله‌ی ۱-مرکز با داده‌های پرت در حالت جویبار داده
۶۰	۴-۲-۲ مسئله‌ی ۱-مرکز پوشاننده در حالت جویبار داده
	۴-۲-۳ مسئله‌ی ۱-مرکز با داده‌های پرت در حالت جویبار داده با تعداد داده‌های
۶۴	پرت ثابت
۶۶	۴-۳ مسئله‌ی ۲-مرکز با داده‌های پرت در حالت جویبار داده
۶۷	۴-۳-۱ حالت $\delta^* \leq \alpha r^*$
۷۲	۴-۳-۲ حالت $\delta^* > \alpha r^*$
۸۷	۵ نتیجه‌گیری
۸۸	۵-۱ کارهای آتی

فهرست شکل‌ها

۱-۱	نمونه‌ای از مسئله‌ی ۲-مرکز	۱۱
۲-۱	نمونه‌ای از مسئله‌ی ۲-مرکز با داده‌های پرت	۱۳
۳-۱	نمونه‌ای از مسئله‌ی ۲-مرکز در حالت پیوسته	۱۴
۱-۳	نمونه‌ای از تخصیص نقاط به ازای مراکز مربع شکل توخالی.	۳۱
۲-۳	نمونه‌ای از تبدیل ورودی مسئله‌ی پوشش رأسی به ورودی مسئله‌ی k -مرکز	۳۲
۳-۳	نمونه‌ای از حل مسئله‌ی ۳-مرکز با الگوریتم گنزالز	۳۳
۴-۳	نمونه‌ای از شبکه‌بندی الگوریتم ضربابی زاده	۳۵
۵-۳	نمونه‌ای از اجرای الگوریتم ضربابی زاده و چن بر روی چهار نقطه	۳۸
۶-۳	اثبات لم ۲-۳	۴۰
۷-۳	اثبات لم ۳-۳	۴۱
۸-۳	نمونه‌ای از تأثیر حذف نقاط پرت در کاهش شعاع مسئله‌ی k -مرکز	۴۲
۱-۴	تعریف فاصله‌ی دو توپ دلخواه	۴۹
۲-۴	اثبات قضیه‌ی ۲-۴	۵۳
۳-۴	گسترش توپ $B(c, r')$ در راستای نقطه‌ی q	۵۵
۴-۴	نحوه‌ی اجرای الگوریتم ۵	۵۹

۶۱	۵-۴ اثبات لم ۶-۴
۶۳	۶-۴ اثبات لم ۷-۴
۶۴	۷-۴ نمودار تابع $\frac{1+\sqrt{\alpha^2-1}}{\alpha}$
۶۷	۸-۴ حالت‌های گراف انتقال
۷۱	۹-۴ اثبات لم ۱۰-۴
۷۲	۱۰-۴ اثبات مشاهده‌ی ۱۲-۴
۷۴	۱۱-۴ اثبات لم ۱۶-۴

فهرست جدول‌ها

۲-۱ نمونه‌هایی از کران پایین تقریب‌پذیری مسائل بهینه‌سازی ۲۴

فصل ۱

مقدمه

مسئله‌ی خوشه‌بندی^۱ یکی از مهم‌ترین مسائل داده‌کاوی^۲ به حساب می‌آید. در این مسئله هدف، دسته‌بندی تعدادی شیء^۳ به گونه‌ای است که اشیاء در یک دسته (خوشه)، نسبت به یکدیگر در برابر دسته‌های دیگر شبیه‌تر باشند (معیارهای متفاوتی برای تشابه تعریف می‌گردد). این مسئله در حوزه‌های مختلفی از علوم کامپیوتر، از جمله داده‌کاوی، جست‌وجوی الگو^۴، پردازش تصویر^۵، بازیابی اطلاعات^۶ و بایوانفورماتیک^۷ مورد استفاده قرار می‌گیرد [۱].

مسئله‌ی خوشه‌بندی، از جمله مسائل مهم علوم کامپیوتر است که مورد توجه بسیاری از دانشمندان قرار گرفته است. راه‌حل‌های الگوریتمی بسیار زیادی برای خوشه‌بندی ارائه شده است. این الگوریتم‌ها را براساس رویکردهای مختلفی که به مسئله دارند، خوشه‌های متفاوتی به دست می‌آورند. در عمل هیچ‌کدام از راه‌حل‌های ارائه شده به طور کلی بر دیگری ارجحیت ندارد و باید راه‌حل مدنظر را متناسب با کاربرد مطرح مورد استفاده قرار داد. به طور مثال استفاده از الگوریتم‌های مرکزگرا، برای خوشه‌های غیر محدب به خوبی عمل نمی‌کند. یکی از رویکردهای شناخته‌شده برای مسئله‌ی خوشه‌بندی، مسئله‌ی k -مرکز است. در این مسئله هدف، پیدا کردن k نقطه به عنوان مرکز دسته‌ها است به طوری که شعاع

^۱Clustering

^۲Data mining

^۳Object

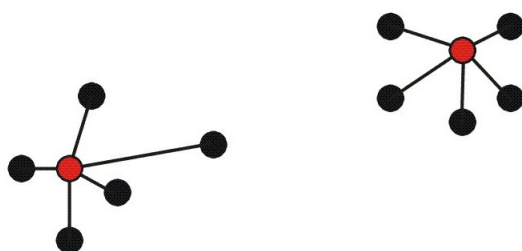
^۴Pattern recognition

^۵Image analysis

^۶Information retrieval

^۷Bioinformatics

دسته‌ها تا حد ممکن کمینه شود. در نظریه‌ی گراف، مسئله‌ی k -مرکز متریک^۸ یا مسئله‌ی مکان‌یابی تسهیلات متریک^۹ یک مسئله‌ی بهینه‌سازی ترکیبیاتی^{۱۰} است.



شکل ۱-۱: نمونه‌ای از مسئله‌ی ۲-مرکز

فرض کنید که n شهر و فاصله‌ی دوبه‌دوی آن‌ها، داده‌شده است. می‌خواهیم k انبار در شهرهای مختلف بسازیم به‌طوری‌که بیش‌ترین فاصله‌ی هر شهری از نزدیک‌ترین انبار به خود، کمینه گردد. در حالت نظریه‌ی گراف آن، این بدان معناست که مجموعه‌ای شامل k رأس انتخاب کنیم به‌طوری‌که بیش‌ترین فاصله‌ی هر نقطه از نزدیک‌ترین نقطه‌اش داخل مجموعه‌ی k عضوی کمینه گردد. توجه نمایید که فاصله‌ی بین رئوس باید در فضای متریک^{۱۱} باشند و یا به زبان دیگر، یک گراف کامل داشته باشیم که فاصله‌ها در آن در رابطه‌ی مثلثی^{۱۲} صدق می‌کنند. مثالی از مسئله‌ی ۲-مرکز در شکل ۱-۱ نشان داده شده است.

در این پژوهش، مسئله‌ی k -مرکز با متریک‌های خاص و برای k های کوچک مورد بررسی قرار گرفته است و هر کدام از جنبه‌های متفاوتی بهبود یافته است. در بخش بعدی، تعریف رسمی^{۱۳} از مسائلی که در این پایان‌نامه مورد بررسی قرار می‌گیرند را بیان نموده و در مورد هر کدام توضیح مختصری می‌دهیم.

^۸Metric

^۹Metric facility location

^{۱۰}Combinatorial optimization

^{۱۱}Metric space

^{۱۲}Triangle equation

^{۱۳}Formal

۱-۱ تعریف مسئله

تعریف دقیق‌تر مسئله k -مرکز در زیر آمده است:

مسئله ۱-۱ (k -مرکز) یک گراف کامل بدون جهت $G = (V, E)$ با تابع فاصله‌ی d ، که از نامساوی مثلثی پیروی می‌کند داده شده است. زیرمجموعه‌ی $S \subseteq V$ با اندازه‌ی k را به گونه‌ای انتخاب کنید که عبارت زیر را کمینه کند:

$$\max_{v \in V} \{ \min_{s \in S} d(v, s) \} \quad (1-1)$$

گونه‌های مختلفی از مسئله k -مرکز با محدودیت‌های متفاوتی به وسیله‌ی پژوهشگران مورد مطالعه قرار گرفته است. از جمله‌ی این گونه‌ها، می‌توان به حالتی که در بین داده‌های ورودی، داده‌های پرت وجود دارد، اشاره کرد. در واقع در این مسئله، قبل از خوشه‌بندی می‌توانیم تعدادی از نقاط ورودی را حذف نموده و سپس به خوشه‌بندی نقاط پردازیم. سختی این مسئله از آنجاست که نه تنها باید مسئله‌ی خوشه‌بندی را حل نمود، بلکه در ابتدا باید تصمیم گرفت که کدام یک از داده‌ها را به عنوان داده‌ی پرت در نظر گرفت که بهترین جواب در زمان خوشه‌بندی به دست آید. در واقع اگر تعداد نقاط پرتی که مجاز به حذف است، برابر صفر باشد، مسئله به مسئله k -مرکز تبدیل می‌شود. نمونه‌ای از مسئله‌ی ۲-مرکز با ۷ داده‌ی پرت را در شکل ۲-۱ می‌توانید ببینید. تعریف دقیق‌تر این مسئله در زیر آمده است:

مسئله ۲-۱ (k -مرکز با داده‌های پرت) یک گراف کامل بدون جهت $G = (V, E)$ با تابع فاصله‌ی d ، که از نامساوی مثلثی پیروی می‌کند داده شده است. زیرمجموعه‌ی $Z \subseteq V$ با اندازه‌ی z و مجموعه‌ی $S \subseteq V - Z$ با اندازه‌ی k را انتخاب کنید به طوری که عبارت زیر را کمینه کند:

$$\max_{v \in V-Z} \{ \min_{s \in S} d(v, s) \} \quad (2-1)$$

گونه‌ی دیگری از مسئله k -مرکز که در سال‌های اخیر مورد توجه قرار گرفته است، حالت جویبار داده‌ی آن است. در این گونه از مسئله k -مرکز، در ابتدا تمام نقاط در دسترس نیستند، بلکه به مرور زمان نقاط در دسترس قرار می‌گیرند. محدودیت دومی که وجود دارد، محدودیت حافظه است، به طوری که نمی‌توان تمام نقاط را در حافظه نگه داشت و بعضاً حتی امکان نگه‌داری در حافظه‌ی جانبی نیز وجود ندارد و به طور معمول باید مرتبه‌ی حافظه‌ای کم‌تر از مرتبه حافظه‌ی خطی^{۱۴} متناسب با تعداد نقاط

^{۱۴}Sublinear



شکل ۱-۲: نمونه‌ای از مسئله‌ی ۲-مرکز با داده‌های پرت

استفاده نمود. از این به بعد به چنین مرتبه‌ای، مرتبه‌ی زیرخطی^{۱۵} می‌گوییم. مدلی که ما در این پژوهش بر روی آن تمرکز داریم مدل جویبار داده تک‌گذره^{۱۶} [۲] است. یعنی تنها یک بار می‌توان از ابتدا تا انتهای داده‌ها را بررسی کرد و پس از عبور از یک داده، اگر آن را در حافظه ذخیره نکرده باشیم، دیگر به آن دسترسی نداریم.

یکی از دغدغه‌هایی که در مسائل جویبار داده وجود دارد، عدم امکان دسترسی به تمام نقاط است. در واقع هم این مشکل وجود دارد که به تمام داده‌های قبلی دسترسی نداریم و هم این مشکل وجود دارد که هیچ اطلاعی از داده‌های آتی نداریم. در نتیجه یکی از تبعات این‌گونه از مسئله‌ی k -مرکز، امکان انتخاب نقطه‌ای به عنوان مرکز برای یک دسته است به‌طوری‌که در بین نقاط ورودی نیست. زیرا از نقاطی که تاکنون آمده‌اند به طور کامل اطلاع نداریم.

تعریف ۱-۱ L_p متریک به ازای دو نقطه‌ی d -بعدی $s(s_1, \dots, s_d)$ و $q(q_1, \dots, q_d)$ ، فاصله‌ی p و q در متریک L_p برابر است با:

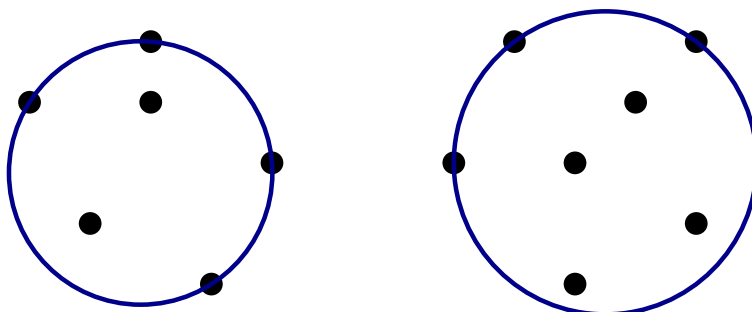
$$d(p, q) = \sqrt[p]{\sum_{i=1}^d (s_i - q_i)^p}$$

این‌گونه از مسئله‌ی k -مرکز، معمولاً تنها برای L_p -متریک مطرح می‌شود یا حالتی که ما مجموعه‌ای از تمام نقاط فضا به انضمام فاصله‌هایشان را داشته باشیم. زیرا مرکز دسته‌ها ممکن است در هر نقطه از فضا قرار بگیرد و ما نیاز داریم که فاصله‌ی آن را از تمام نقاط بدانیم. نمونه‌ای از مسئله‌ی ۲-مرکز

^{۱۵}sublinear

^{۱۶}Single pass

در حالت پیوسته، در شکل ۱-۳ نشان داده شده است. تعریف دقیق گونه‌ی جویبار داده‌ی مسئله‌ی k -مرکز، در زیر آمده است:



شکل ۱-۳: نمونه‌ای از مسئله‌ی ۲-مرکز در حالت پیوسته

مسئله‌ی ۱-۳ (k -مرکز در حالت جویبار داده) مجموعه‌ی U از نقاط فضای d -بعدی داده شده است. زیرمجموعه $S \subseteq U$ با اندازه‌ی k را انتخاب کنید به طوری که عبارت زیر کمینه شود:

$$\max_{u \in U} \{ \min_{s \in S} L_p(u, s) \} \quad (1-3)$$

از آنجایی که گونه‌ی جویبار داده و داده پرت مسئله‌ی k -مرکز به علت به روز بودن مبحث داده‌های حجیم^{۱۷}، به تازگی مورد توجه قرار گرفته است. در این تحقیق سعی شده است که تمرکز بر روی این گونه‌ی خاص از مسئله باشد. همچنین در این پژوهش سعی می‌شود گونه‌های مسئله را برای انواع متریک‌ها و برای k های کوچک نیز مورد بررسی قرار داد.

۲-۱ اهمیت موضوع

مسئله‌ی k -مرکز و گونه‌های آن کاربردهای زیادی در داده‌کاوی دارند. این الگوریتم یکی از رایج‌ترین الگوریتم‌های مورد استفاده برای خوشه‌بندی محسوب می‌شود. به علت افزایش حجم داده‌ها و تولید داده‌ها در طول زمان، گونه‌ی جویبار داده‌ی مسئله در سال‌های اخیر مورد توجه قرار گرفته است. از طرفی مسئله‌ی k -مرکز در بعضی مسائل مانند ارسال نامه از تعدادی مراکز پستی به گیرنده‌ها، نیاز دارد که تمام نامه‌ها را به دست گیرنده‌ها برساند و در نتیجه باید همه‌ی نقاط را پوشش دهد، ولی برای بعضی از مسائل تجاری، نیازی به پوشش تمام نقاط هدف نیست و از لحاظ اقتصادی به صرفه است که نقاط پرت

^{۱۷}Big data

در نظر گرفته نشود. به طور مثال، شرکت Kmart در سال ۲۰۱۰ اعلام کرده است که به ۸۸ درصد جمعیت آمریکا با فاصله‌ای حداکثر ۶ مایل، می‌تواند سرویس دهد. در صورتی که اگر این شرکت قصد داشت تمام جمعیت آمریکا را پوشش دهد، نیاز داشت تعداد شعب یا شعاع پوشش خود را به میزان قابل توجهی افزایش دهد که از لحاظ اقتصادی به صرفه نبود [۳]. علاوه بر دو گونه‌ی مطرح شده در این قسمت، گونه‌های دیگر از مسئله‌ی k -مرکز در مرجع [۳] آمده است.

۱-۳ ادبیات موضوع

همان‌طور که ذکر شد مسئله‌ی k -مرکز در حالت داده‌های پرت و جویبار داده، گونه‌های تعمیم‌یافته از مسئله‌ی k -مرکز هستند و در حالت‌های خاص به مسئله‌ی k -مرکز کاهش پیدا می‌کنند. مسئله‌ی k -مرکز در حوزه‌ی مسائل ان‌پی-سخت^{۱۸} قرار می‌گیرد و با فرض $P \neq NP$ الگوریتم دقیق با زمان چندجمله‌ای برای آن وجود ندارد [۴]. بنابراین برای حل کارای^{۱۹} این مسائل از الگوریتم‌های تقریبی^{۲۰} استفاده می‌شود.

برای مسئله‌ی k -مرکز، دو الگوریتم تقریبی معروف وجود دارد. در الگوریتم اول، که به روش حریصانه^{۲۱} عمل می‌کند، در هر مرحله بهترین مرکز ممکن را انتخاب می‌کند به طوری تا حد ممکن از مراکز قبلی دور باشد [۵]. این الگوریتم، الگوریتم تقریبی با ضریب تقریب ۲ ارائه می‌دهد. در الگوریتم دوم، با استفاده از مجموعه‌ی غالب کمینه^{۲۲}، الگوریتمی با ضریب تقریب ۲ ارائه می‌گردد [۶]. همچنین ثابت شده است، که بهتر از این ضریب تقریب، الگوریتمی نمی‌توان ارائه داد مگر آن‌که $P = NP$ باشد.

برای مسئله‌ی k -مرکز در حالت جویبار داده برای ابعاد بالا، بهترین الگوریتم موجود ضریب تقریب $2 + \epsilon$ دارد [۷، ۸، ۹] و ثابت می‌شود الگوریتمی با ضریب تقریب بهتر از ۲ نمی‌توان ارائه داد. برای مسئله‌ی k -مرکز با داده‌ی پرت در حالت جویبار داده نیز، بهترین الگوریتم ارائه شده، الگوریتمی با ضریب تقریب $4 + \epsilon$ است که با کران پایین ۳ هنوز اختلاف قابل توجهی دارد [۳].

^{۱۸}NP-hard

^{۱۹}Efficient

^{۲۰}Approximation algorithm

^{۲۱}Greedy

^{۲۲}Dominating set

برای k های کوچک به خصوص، $k = 1, 2$ ، الگوریتم های بهتری ارائه شده است. بهترین الگوریتم ارائه شده برای مسئله ی ۱ - مرکز در حالت جویبار داده برای ابعاد بالا، دارای ضریب تقریب $1/22$ است و کران پایین $\frac{1+\sqrt{2}}{3}$ نیز برای این مسئله اثبات شده است [۱۰، ۱۱]. برای مسئله ۲ - مرکز در حالت جویبار داده برای ابعاد بالا، اخیراً راه حلی با ضریب تقریب $1/8 + \epsilon$ ارائه شده است [۱۲]. برای مسئله ی ۱ - مرکز با داده ی پرت، تنها الگوریتم موجود، الگوریتمی با ضریب تقریب $1/73$ است [۱۳].

۴-۱ اهداف تحقیق

در این پایان نامه مسئله ی k - مرکز در حالت جویبار داده با داده های پرت در حالت های مختلف مورد بررسی قرار می گیرد و سعی خواهد شد که نتایج قبلی در این مسائل را از جنبه های مختلفی مورد بهبود دهد.

اولین مسئله ای که مورد بررسی قرار گرفته است، ارائه الگوریتمی تقریبی، برای مسئله ی ۱ - مرکز در حالت جویبار داده است به طوری که، نه تنها تمام نقاط ورودی را می پوشاند بلکه تضمین می کند که دایره ی بهینه ی جواب ۱ - مرکز برای نقاط ورودی را نیز پوشاند. در تلاش اول، الگوریتم جدیدی با حافظه و زمان به روزرسانی $O(\frac{d}{\epsilon})$ و با ضریب تقریب $1/8 + \epsilon$ ، برای این مسئله ارائه می گردد. با بررسی های بیش تر، الگوریتم دیگری با ضریب تقریب $1/7$ ، با الگوریتمی کاملاً متفاوت، با حافظه ی $O(d)$ برای این مسئله ارائه می گردد.

مسئله ی دومی که مورد بررسی قرار می گیرد، مسئله ی ۱ - مرکز در حالت جویبار داده با داده های پرت است. در ابتدا الگوریتمی ساده با حافظه ی $O(zd)$ و زمان به روزرسانی $O(d + \log(z))$ با ضریب تقریب ۲ برای این مسئله ارائه می گردد. در بررسی های بعدی، با استفاده از نتایج به دست آمده در قسمت قبل، برای z های کوچک، الگوریتمی با حافظه ی $O(dz^{d+1})$ با ضریب تقریب $1/7$ برای این مسئله ارائه شد که ضریب تقریب بهترین الگوریتم موجود را، از $1/73$ به $1/7$ کاهش می دهد.

مسئله ی سومی که مورد بررسی قرار گرفت، مسئله ی ۲ - مرکز در حالت جویبار داده با داده های پرت است. بهترین الگوریتمی که در حال حاضر برای این مسئله وجود دارد، یک الگوریتم با ضریب تقریب $4 + \epsilon$ است [۷]. ما با ارائه ی الگوریتمی جدید، الگوریتمی با ضریب تقریب $1/8 + \epsilon$ برای این مسئله ارائه دادیم که بهبودی قابل توجه محسوب می شود.

۵-۱ ساختار پایان نامه

این پایان نامه در پنج فصل به شرح زیر ارائه خواهد شد. در فصل دوم به بیان مفاهیم و تعاریف مرتبط با موضوعات مورد بررسی در بخش های دیگر خواهیم پرداخت. سعی شده، تا حد امکان با زبان ساده و ارجاع های مناسب، پایه ی لازم را برای فصول بعدی در این فصل فراهم آوریم. فصل سوم این پایان نامه شامل مطالعه و بررسی کارهای پیشین انجام شده مرتبط با موضوع این پایان نامه خواهد بود. این فصل در سه بخش تنظیم گردیده است. در بخش اول، مسئله ی k -مرکز در حالت ایستا مورد بررسی قرار می گیرد. در بخش دوم، حالت جویبار داده ی مسئله و مجموعه هسته های^{۲۳} مطرح برای این مسئله مورد بررسی قرار می گیرد. در نهایت، در بخش سوم، مسئله ی k -مرکز با داده های پرت مورد بررسی قرار می گیرد.

در فصل چهارم، نتایج جدیدی که در این پایان نامه به دست آمده است، ارائه می شود. این نتایج شامل الگوریتم های جدید برای مسئله ی ۱-مرکز در حالت جویبار داده، مسئله ی ۱-مرکز با داده های پرت در حالت جویبار داده و مسئله ی دو مرکز در حالت جویبار داده با داده های پرت می شود.

در فصل پنجم، به جمع بندی کارهای انجام شده در این پژوهش و ارائه ی پیشنهادهایی برای انجام کارهای آتی و تعمیم هایی که از راه حل ارائه شده وجود دارد، خواهیم پرداخت.

^{۲۳} Coreset

فصل ۲

مفاهیم اولیه

در این فصل به تعریف و بیان مفاهیم پایه‌ای مورد استفاده در فصل‌های بعد می‌پردازیم. با توجه به مطالب مورد نیاز در فصل‌های آتی، مطالب این فصل به چهار بخش، مسائل ان‌پی-سخت، مسائل خوشه‌بندی، الگوریتم‌های تقریبی و الگوریتم‌های جویبار داده تقسیم می‌شود.

۲-۱ مسائل ان‌پی-سخت

یکی از اولین سؤال‌های بنیادی مطرح در علم کامپیوتر، اثبات عدم حل‌پذیری بعضی از مسائل است. به عنوان نمونه، می‌توان از دهمین مسئله‌ی هیلبرت^۱ در کنگره‌ی ریاضی یاد کرد. هیلبرت این مسئله را این‌گونه بیان کرد: ”فرآیندی طراحی کنید که در تعداد متناهی گام بررسی کند که آیا یک چندجمله‌ای، ریشه‌ی صحیح^۲ دارد یا خیر“. با مدل محاسباتی که به وسیله‌ی تورینگ^۳ ارائه شد، این مسئله معادل پیدا کردن الگوریتمی برای این مسئله است که اثبات می‌شود امکان‌پذیر نیست [۱۴]. برخلاف مثال بالا، عمده‌ی مسائل علوم کامپیوتر از نوع بالا نیستند و برای طیف وسیعی از آن‌ها، الگوریتم‌های پایان‌پذیر وجود دارد. بیشتر تمرکز علوم کامپیوتر هم بر روی چنین مسائلی است.

اگر چه برای اکثر مسائل، الگوریتمی پایان‌پذیر وجود دارد، اما وجود چنین الگوریتمی لزومی بر حل

^۱Hilbert

^۲Integral root

^۳Turing

شدن چنین مسائلی نیست. در عمل، علاوه بر وجود الگوریتم، میزان کارآمدی^۴ الگوریتم نیز مطرح می‌گردد. به طور مثال، اگر الگوریتم حل یک مسئله مرتبه‌ی بالا یا نمایی داشته باشد، الگوریتم ارائه‌شده برای آن مسئله برای ورودی‌های نسبتاً بزرگ قابل اجرا نیست و نمی‌توان از آن برای حل مسئله استفاده کرد. برای تشخیص و تمیز کارآمدی الگوریتم‌های مختلف و همچنین میزان سختی مسائل در امکان ارائه‌ی الگوریتم‌های کارآمد یا غیر کارآمد، نظریه‌ی پیچیدگی^۵، دسته‌بندی‌های مختلفی برای سختی مسائل و حل‌پذیری آن‌ها ارائه داده است تا بتوان به‌طور رسمی^۶ در مورد این معیارها صحبت کرد. برای دسته‌بندی مسائل در نظریه‌ی پیچیدگی، ابتدا آن‌ها را به صورت تصمیم‌پذیر بیان می‌کنند.

مسئله‌ی ۱-۲ (مسائل تصمیم‌گیری)^۷ به دسته‌ای از مسائل گفته می‌شود که پاسخ آن‌ها تنها بله یا خیر است.

به عنوان مثال، اگر بخواهیم مسئله‌ی ۱- مرکز در فضای \mathbb{R}^d را به صورت تصمیم‌پذیر بیان کنیم، به مسئله‌ی زیر می‌رسیم:

مسئله‌ی ۲-۲ (نسخه‌ی تصمیم‌پذیر ۱- مرکز) مجموعه‌ی نقاط در فضا \mathbb{R}^d و شعاع r داده شده است، آیا دایره‌ای به شعاع r وجود دارد که تمام نقاط را بپوشاند؟

در نظریه‌ی پیچیدگی، می‌توان گفت عمده‌ترین دسته‌بندی موجود، دسته‌بندی مسائل تصمیم‌گیری به مسائل پی (P) و ان‌پی (NP) است. رده‌ی مسائل P، شامل تمامی مسائل تصمیم‌گیری است که راه‌حل چندجمله‌ای برای آن‌ها وجود دارد. از طرفی رده‌ی مسائل NP، شامل تمامی مسائل تصمیم‌گیری است که در زمان چندجمله‌ای قابل صحت‌سنجی^۸ اند. تعریف صحت‌سنجی در نظریه پیچیدگی، یعنی اگر جواب مسئله‌ی تصمیم‌گیری بله باشد، می‌توان اطلاعات اضافی با طول چندجمله‌ای ارائه داد، که در زمان چندجمله‌ای از روی آن، می‌توان جواب بله الگوریتم را تصدیق نمود. به طور مثال، برای مسئله‌ی ۱- مرکز، کافی است به عنوان تصدیق جواب بله، مرکز دایره‌ی پوشاننده، ارائه داده شود. در این صورت، می‌توان با مرتبه‌ی خطی بررسی نمود که تمام نقاط داخل این دایره قرار می‌گیرند یا نه. برای مطالعه‌ی بیش‌تر و تعاریف دقیق‌تر می‌توان به مرجع [۱۴] مراجعه نمود.

^۴Efficiency

^۵Complexity theory

^۶Formal

^۷Decision problems

^۸Verifiable

همان‌طور که می‌دانید درستی یا عدم درستی $P \subset NP$ از جمله معروف‌ترین مسائل حل‌نشده^۹ در نظریه پیچیدگی است. حدس بسیار قوی وجود دارد که $P \neq NP$ و بسیاری از مسائل، با این فرض حل می‌شوند و در صورتی که زمانی، خلاف این فرض اثبات گردد، آنگاه قسمت عمده‌ای از علوم کامپیوتر زیر سؤال می‌رود.

در نظریه پیچیدگی، برای دسته‌بندی مسائل، یکی از روش‌های دسته‌بندی کاهش چندجمله‌ای^{۱۰} مسائل به یکدیگر است.

تعریف ۱-۲ می‌گوییم مسئله‌ای A در زمان چندجمله‌ای به مسئله‌ی B کاهش می‌یابد، اگر وجود داشته باشد الگوریتم چندجمله‌ای C که به ازای هر ورودی α برای مسئله‌ی A ، یک ورودی β در زمان چندجمله‌ای برای مسئله‌ی B بسازد، به‌طوری‌که A ، α را می‌پذیرد اگر و تنها اگر B ، β را بپذیرد. در اینجا منظور از پذیرفتن جواب بله به ورودی است.

از این به بعد برای سادگی به جای کاهش چندجمله‌ای از واژه‌ی کاهش استفاده می‌کنیم. در پی جستجوهای که برای برابری دسته‌ی پی و ان‌پی صورت گرفت، مجموعه‌ای از مسائل که عمدتاً داخل ان‌پی هستند استخراج گردید که اگر ثابت شود یکی از آن‌ها متعلق به پی است، آنگاه تمام مسائل دسته‌ی ان‌پی متعلق به پی خواهند بود و در نتیجه $P = NP$ می‌گردد. به این مجموعه مسائل ان‌پی-سخت می‌گویند. در واقع مسائل این دسته، مسائلی هستند که تمام مسائل داخل دسته‌ی ان‌پی، به آن‌ها کاهش می‌یابند.

کوک و لوین در قضیه‌ای به نام قضیه‌ی کوک-لوین ثابت کردند مسئله‌ی صدق‌پذیری^{۱۱} یک مسئله‌ی ان‌پی-سخت است [۱۴]. با پایه قرار دادن این اثبات و استفاده از تکنیک کاهش، اثبات ان‌پی-سخت بودن سایر مسائل، بسیار ساده‌تر گردید.

۲-۲ مسائل خوشه‌بندی

همان‌طور که در مقدمه گفته شد، خوشه‌بندی یکی از مسائل پرکاربرد علوم کامپیوتر محسوب می‌شود. مسئله‌ی خوشه‌بندی دارای یک الگوریتم واحد نیست. تا کنون راه‌حل‌های زیادی برای این مسئله ارائه

^۹Open problem

^{۱۰}Polynomial Reduction

^{۱۱}Satisfiability problem

شده است که از لحاظ معیار تشخیص خوشه‌ها و نحوه‌ی انتخاب یک خوشه، با یک‌دیگر تفاوت بسیاری دارند. به همین خاطر مسئله‌ی خوشه‌بندی یک مسئله‌ی بهینه‌سازی چندهدفه^{۱۲} محسوب می‌شود.

برای این‌که بتوانیم برای یک مجموعه داده^{۱۳}، یک الگوریتم خوب برای خوشه‌بندی انتخاب کنیم، نیاز داریم که علاوه بر انتخاب الگوریتم مناسب متناسب با مجموعه داده، متغیرهای^{۱۴} دیگری نیز باید انتخاب شوند. از جمله مهم‌ترین متغیرهای مطرح در الگوریتم‌های ارائه شده عبارت‌اند از:

- تابع فاصله‌ی بین داده‌ها

- آستانه‌ی^{۱۵} تراکم^{۱۶}

- تعداد خوشه‌ها

هر یک از این متغیرها باید متناسب با مجموعه داده و کاربردی که از نتیجه‌ی الگوریتم مد نظر است، تنظیم گردد. با توجه به پیچیدگی موجود، خوشه‌بندی معمولاً یک عمل تکرارشونده‌ی^{۱۷} است که با سعی و خطا سعی در بهینه‌سازی چندهدفه موجود دارد. بنابراین برای اینکه به جواب رضایت بخش برسیم، نیاز است که بارها الگوریتم و متغیرهای موجود را تغییر داد.

همان‌طور که در مرجع [۱۵] ذکر شده است، خوشه در خوشه‌بندی تعریف واحدی ندارد و یکی از دلایل وجود الگوریتم‌های متفاوت، همین تفاوت تعریف‌ها از خوشه است. بنابراین با توجه به مدلی که برای خوشه‌ها ارائه می‌شود، الگوریتم متفاوتی نیز ارائه می‌گردد. در ادامه به بررسی تعدادی از معروف‌ترین مدل‌های مطرح می‌پردازیم:

- **مدل‌های مرکزگرا:** در این مدل‌ها، هر دسته با یک مرکز نشان داده می‌شود. از جمله معروف‌ترین روش‌های خوشه‌بندی بر اساس این مدل، خوشه‌بندی K - میانگین^{۱۸} و خوشه‌بندی K - میانه^{۱۹} و خوشه‌بندی K - مرکز است.

^{۱۲}Multi-objective

^{۱۳}Data set

^{۱۴}Parameter

^{۱۵}Threshold

^{۱۶}Density

^{۱۷}Iterative

^{۱۸} K -Means

^{۱۹} K -Median

• **مدل‌های مبتنی بر توزیع نقاط:** در این مدل، دسته‌ها با فرض پیروی از یک توزیع احتمالی مشخص می‌شوند. از جمله الگوریتم‌های معروف ارائه شده در این مدل، الگوریتم بیشینه‌سازی امید ریاضی^{۲۰} است.

• **مدل‌های مبتنی بر تراکم نقاط:** در این مدل، خوشه‌ها متناسب با ناحیه‌های متراکم نقاط در مجموعه داده مورد استفاده قرار می‌گیرد.

• **مدل‌های مبتنی بر گراف:** در این مدل، هر خوشه به مجموعه از رئوس گفته می‌شود که تمام رئوس آن با یک‌دیگر همسایه باشند. از جمله الگوریتم‌های معروف این مدل، الگوریتم خوشه‌بندی HCS^{۲۱} است.

الگوریتم‌های ارائه شده تنها از نظر نوع مدل با یک‌دیگر متفاوت نیستند. بلکه، می‌توان آن‌ها را از لحاظ نحوه‌ی تخصیص نقاط بین خوشه‌ها نیز تقسیم‌بندی کرد:

• **تخصیص قطعی داده‌ها:** در این نوع خوشه‌بندی هر داده دقیقاً به یک خوشه اختصاص داده می‌شود.

• **تخصیص قطعی داده‌ها با داده‌ی پرت:** در این نوع خوشه‌بندی ممکن است بعضی از داده‌ها به هیچ خوشه‌ای اختصاص نیابد، اما بقیه داده‌ها هر کدام دقیقاً به یک خوشه اختصاص می‌یابد.

• **تخصیص قطعی داده:** در این نوع خوشه‌بندی هر داده دقیقاً به یک خوشه اختصاص داده می‌شود.

• **خوشه‌بندی هم‌پوشان:** در این نوع خوشه‌بندی هر داده می‌تواند به چند خوشه اختصاص داده شود. در گونه‌ای از این مدل، می‌توان هر نقطه را با احتمالی به هر خوشه اختصاص می‌یابد. به این گونه از خوشه‌بندی، خوشه‌بندی نرم^{۲۲} گفته می‌شود.

• **خوشه‌بندی سلسه‌مراتبی:** در این نوع خوشه‌ها، داده‌ها به گونه‌ای به خوشه‌ها تخصیص داده می‌شود که دو خوشه یا اشتراک ندارند یا یکی به طور کامل دیگری را می‌پوشاند. در واقع در بین خوشه‌ها، رابطه‌ی پدر فرزندی برقرار است.

^{۲۰} Expectation-maximization

^{۲۱} Highly Connected Subgraphs

^{۲۲} Soft clustering

در بین دسته‌بندی‌های ذکر شده، تمرکز اصلی این پایان‌نامه بر روی مدل مرکزگرا و خوشه‌بندی قطعی با داده‌های پرت با مدل k -مرکز است. همان‌طور که ذکر شد علاوه بر مسئله‌ی k -مرکز که به تفصیل مورد بررسی قرار می‌گیرد، k -میانه و k -میانگین از جمله معروف‌ترین خوشه‌بندی‌های مدل مرکزگرا هستند. در خوشه‌بندی k -میانه، هدف افراز نقاط به k خوشه است به گونه‌ای که مجموع مربع فاصله‌ی هر نقطه از میانه‌ی نقاط آن خوشه، کمینه گردد. در خوشه‌بندی k -میانگین، هدف افراز نقاط به k خوشه است به گونه‌ای که مجموع فاصله‌ی هر نقطه از میانگین نقاط داخل خوشه (یا مرکز آن خوشه) کمینه گردد.

۲-۳ الگوریتم‌های تقریبی

تا اینجا با رده‌بندی مسائل به دو دسته‌ی پی و ان‌پی آشنا شدیم. نه تنها مسائل ان‌پی، بلکه بعضی از مسائل پی نیز دارای الگوریتم کارآمدی نیستند. در عمل، یک الگوریتم چندجمله‌ای با مرتبه‌ی بالا (بیش از ۴)، یک الگوریتم کارآمد محسوب نمی‌شود، زیرا نمی‌توان با کامپیوترهای عادی امروزی، برای n های بزرگ‌تر از ۱۰۰۰ در زمانی کم‌تر از ۱ دقیقه به پاسخ رسید. با این تعریف، به‌طور مثال هنوز الگوریتم کارآمدی برای فهمیدن این‌که یک عدد اول است یا نه پیدا نشده است، با اینکه الگوریتم ارائه شده یک الگوریتم چندجمله‌ای است. عمده‌ی مسائل کاربردی مطرح در دنیای واقع، یا متعلق به دسته‌ی ان‌پی-سخت هستند و در نتیجه راه‌حل چندجمله‌ای ندارند، یا اگر راه‌حل چندجمله‌ای داشته باشند، مرتبه‌ی چندجمله‌ای ارائه شده بالاست و در نتیجه راه‌حل کارآمدی محسوب نمی‌گردد. یکی از رویکردهای رایج در برابر چنین مسائلی، صرف نظر کردن از دقت راه‌حل‌هاست. به‌طور مثال راه‌حل‌های ابتکاری^{۲۳} گوناگونی برای مسائل مختلف به خصوص مسائل ان‌پی-سخت بیان شده است. این راه‌حل‌ها بدون این‌که تضمین کنند راه‌حل خوبی ارائه می‌دهند یا حتی جوابشان به جواب بهینه نزدیک است، اما با معیارهایی سعی در بهینه عمل کردن دارند و تا حد ممکن سعی در ارائه‌ی جواب بهینه یا نزدیک بهینه دارند. اما در عمل، تنها برای دسته‌ای از کاربردها پاسخ قابل قبولی ارائه می‌دهند.

مشکل عمده‌ی راه‌حل‌های ابتکاری، عدم امکان استفاده از آن‌ها برای تمام کاربردها است. بنابراین در رویکرد دوم که اخیراً نیز مطرح شده است، سعی در ارائه‌ی الگوریتم‌های ابتکاری شده است که تضمین می‌کنند اختلاف زیادی با الگوریتمی که جواب بهینه می‌دهد، نداشته باشند. در واقع این الگوریتم‌ها

^{۲۳}Heuristic

مسئله	کران پایین تقریب پذیری
k - مرکز	$2[6]$
k - مرکز در فضای اقلیدسی	$1/822[16]$
۱ - مرکز در حالت جویبار داده	$1+\sqrt{2}[10]$
k - مرکز با نقاط پرت و نقاط اجباری	$3[3]$

جدول ۲-۱: نمونه‌هایی از کران پایین تقریب‌پذیری مسائل بهینه‌سازی

همواره و در هر شرایطی، تقریبی از جواب بهینه را ارائه می‌دهند. به چنین الگوریتم‌هایی، الگوریتم‌های تقریبی^{۲۴} می‌گویند. علت اصلی این نام‌گذاری، تقریب زدن جواب الگوریتم بهینه است. ضریب تقریب یک الگوریتم تقریبی، به حداکثر نسبت جواب الگوریتم تقریبی به جواب بهینه گفته می‌شود.

الگوریتم‌های تقریبی تنها به علت محدودیت کارایی الگوریتم‌هایی که جواب بهینه می‌دهند، مورد استفاده قرار نمی‌گیرند. هر نوع محدودیتی ممکن است، استفاده از الگوریتم تقریبی را نسبت به الگوریتمی که جواب بهینه می‌دهد، مقرون به صرفه کند. به طور مثال از جمله عوامل دیگری که ممکن است باعث این انتخاب شود، کاهش میزان حافظه‌ی مصرفی باشد. برای طیف وسیعی از مسائل، کمبود حافظه، باعث می‌شود الگوریتم‌هایی با حافظه‌ی مصرفی کم‌تر طراحی شود که به دقت الگوریتم‌های بهینه عمل نمی‌کند اما می‌تواند با مصرف کم‌تر به تقریبی از جواب بهینه دست یابد. معمولاً چنین الگوریتم‌هایی حافظه‌ی مصرفی از مرتبه‌ی زیرخطی^{۲۵} دارند و به همین دلیل برای داده‌های حجیم بسیار کاربرد دارند.

۲-۳-۱ میزان تقریب‌پذیری مسائل

همان‌طور که تا اینجا دیدیم، یکی از راه‌کارهایی که برای کارآمد کردن راه‌حل ارائه شده برای یک مسئله وجود دارد، استفاده از الگوریتم‌های تقریبی برای حل آن مسئله است. یکی از عمده‌ترین دغدغه‌های مطرح در الگوریتم‌های تقریبی کاهش ضریب تقریب است. در بعضی از موارد حتی امکان ارائه‌ی الگوریتم تقریبی با ضریبی ثابت نیز وجود ندارد. به طور مثال، همان‌طور که در فصل کارهای پیشین بیان خواهد شد، الگوریتم تقریبی با ضریب تقریب کم‌تر از ۲، برای مسئله‌ی k - مرکز وجود ندارد مگر

^{۲۴} Approximation Algorithm

^{۲۵} Sublinear

اینکه $P = NP$ باشد. برای مسائل مختلف، معمولاً می‌توان کران پایینی برای میزان تقریب‌پذیری آنها ارائه داد. در واقع برای برخی مسائل ان‌پی-سخت، علاوه بر این که الگوریتم کارآمدی وجود ندارد، بعضاً الگوریتم تقریبی با ضریبی تقریب کم و نزدیک به یک نیز وجود ندارد. در جدول ۲-۱ میزان تقریب‌پذیری مسائل مختلفی که در این پایان‌نامه مورد استفاده قرار می‌گیرد را می‌بینید.

۲-۴ الگوریتم‌های جویبار داده

در علوم کامپیوتر، کاربردهایی وجود دارد که تمام داده‌ی ورودی در لحظه‌ی شروع الگوریتم در دسترس نیست. بنابراین مدل جدیدی برای این الگوریتم‌ها ارائه می‌شود که در آن ورودی به مرور زمان در اختیار الگوریتم قرار می‌گیرد. به علت حجم زیاد داده‌ها، معمولاً امکان ذخیره‌سازی داده‌ها برای استفاده‌های بعدی وجود ندارد و در نتیجه، در این الگوریتم‌ها تنها می‌توان چند بار (معمولاً یک بار) از ابتدای ورودی تا انتهای ورودی، به داده‌ها دسترسی داشت. در واقع الگوریتم‌های جویبار داده، به الگوریتم‌هایی گفته می‌شوند که ورودی آن‌ها یک یا چند دنباله است که الگوریتم می‌تواند به ترتیب دنباله، یک یا چند بار از ابتدای دنباله تا انتهای آن، به اعضای دنباله دسترسی داشته باشد.

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

در سال‌های اخیر، پیشرفت‌های حوزه‌ی تکنولوژی، امکان جمع‌آوری داده‌ها را به صورت پیوسته ممکن ساخته است. به طور مثال می‌توان تراکنش‌های بانکی، استفاده از تلفن همراه یا مرورگر وب خود را در نظر بگیرید. در هر تعاملی که با این سیستم‌ها انجام می‌دهید، میزان زیادی داده تولید شده و ذخیره می‌گردد. در اکثر مواقع می‌توان این حجم عظیم داده را مورد پردازش قرار داد و اطلاعات بسیار مفیدی از آن، استخراج نمود. زمانی که حجم داده بسیار زیاد باشد، معمولاً چالش‌های محاسباتی و الگوریتمی برای الگوریتم‌ها به وجود می‌آورد. از جمله‌ی آن‌ها می‌توان به موارد زیر اشاره کرد:

- با افزایش حجم داده، امکان پردازش داده‌ها به صورت کارآمد با چند بار عبور کردن از جویبار داده وجود ندارد. یکی از مهم‌ترین موانع در طراحی الگوریتم‌های کارآمد برای مدل جویبار داده این

است که الگوریتم‌ها مجبور هستند هر داده را حداکثر یک بار مورد پردازش قرار دهند. بنابراین الگوریتم‌های مدل جویبار داده، معمولاً تک‌گذره‌اند.

- در اکثر الگوریتم‌های جویبار داده، از یک واحد برای پردازش داده‌ها به صورت محلی استفاده می‌شود. علت اصلی وجود چنین واحدی، نیاز این الگوریتم‌ها به تشخیص تغییرات در داده‌های ورودی است. این عملکرد الگوریتم‌های جویبار داده را می‌توان نوعی استفاده از اصل محل‌گرایی^{۲۶} به حساب آورد. اما مشکل عمده‌ی این نوع رویکرد اجتناب‌ناپذیر، در عمده‌ی موارد، عدم امکان ارائه‌ی راه‌حلی مناسب برای مسئله است. در ساخت الگوریتم‌های جویبار داده، باید دقت و تمرکز عمده‌ای را صرف تشخیص نحوه‌ی تغییر داده‌های ورودی نمود.

- یکی از مهم‌ترین مشخصه‌های جویبار داده‌ها، ماهیت عدم متمرکز بودن داده‌ها است. از طرفی یک پردازنده به تنهایی دارای محدودیت بسیار زیادی از نظر قدرت پردازشی و حافظه است. بنابراین معمولاً الگوریتم‌های جویبار داده، به‌گونه‌ای طراحی می‌شوند که توزیع‌پذیر بوده و توانایی اجرا شدن به صورت چندروندی^{۲۷} را دارا باشند.

۲-۴-۱ گونه‌های مطرح

در مدل جویبار داده، بعضی یا همه‌ی نقاط ورودی که باید پردازش گردند برای دسترسی تصادفی^{۲۸} از حافظه‌ی اصلی یا حافظه‌ی جانبی در دسترس نیستند، بلکه به مرور زمان، به صورت جویبار یا جویبارهایی از داده در اختیار الگوریتم قرار می‌گیرند [۲].

هر جویبار را می‌توان به عنوان دنباله‌ای مرتب از نقاط (یا به‌روزرسانی‌ها^{۲۹}) در نظر گرفت به طوری که به ترتیب دنباله قابل دسترسی هستند و هر کدام را تنها به تعدادی محدود بار (معمولاً یک بار) می‌توان خواند [۲].

مسائل زیادی در این مدل مورد بررسی قرار گرفته‌اند که از جمله مهم‌ترین آن‌ها می‌توان به موارد زیر اشاره کرد:

^{۲۶}Locality

^{۲۷}Multi Thread

^{۲۸}Random access

^{۲۹}Update

- محاسبه‌ی آماری مربوط به توزیع داده‌های یک جویبار داده به قدری بزرگ که قابل ذخیره شدن نیست.
- مسائل مربوط به گراف‌ها، به‌طوری‌که ماتریس مجاورت گراف به ترتیب دلخواهی به صورت جویبار داده به الگوریتم داده می‌شود.
- مسائل مربوط به دنباله‌ها و استخراج اطلاعات از دنباله‌ها که وابستگی شدیدی به ترتیب اعضای دنباله دارند، مانند پیدا کردن طولانی‌ترین زیردنباله با اعضای صعودی یا پیدا کردن تعداد نابه‌جایی‌های^{۳۰} داخل دنباله. [۲]

۲-۴-۲ تحلیل الگوریتم‌های جویبار داده

کارایی یک الگوریتم جویبار داده بر اساس سه معیار زیر اندازه‌گیری می‌شود:

- تعداد مرتبه‌ای که الگوریتم از روی جویبار داده گذر می‌کند
- میزان حافظه‌ی مصرفی
- زمان مصرفی به ازای پردازش هر داده

الگوریتم‌های جویبار داده تشابه‌های زیادی با الگوریتم‌های برخط^{۳۱} دارند. به طور مثال، هر دو نوع الگوریتم، نیاز دارند قبل از اینکه تمام ورودی فرا برسد، در مورد داده‌ی جویبار داده تصمیم بگیرند. اما تفاوت‌های عمده‌ای نیز بین این الگوریتم‌ها وجود دارد. الگوریتم‌های جویبار داده، دارای حافظه‌ی محدودی هستند، اما می‌توانند تصمیم‌گیری راجع به یک داده را تا چند مرحله به عقب بیاندازند، در صورتی که الگوریتم‌های برخط، به محض ورود داده، باید در مورد آن تصمیم بگیرند.

۲-۴-۳ مجموعه هسته

یکی از تکنیک‌های رایج در الگوریتم‌های جویبار داده، نگهداری نماینده‌ای با اندازه‌ی بسیار کوچک‌تر نسبت به اندازه‌ی جویبار داده است. این مجموعه معمولاً دغدغه‌ی محدودیت حافظه را در الگوریتم‌های

^{۳۰} Inversion

^{۳۱} Online

جویبار داده برطرف می‌کند. به چنین مجموعه‌ای، مجموعه هسته می‌گوییم. حال به تعریف رسمی مجموعه هسته می‌پردازیم:

تعریف ۲-۲ فرض کنید μ یک تابع اندازه‌گیری^{۳۲} (همانند تابع عرض مجموعه‌ای از نقاط) از \mathbb{R}^d به اعداد حقیقی نامنفی $\{0\} \cup \mathbb{R}^+$ باشد. فرض کنید که این تابع، یک تابع یکنواخت است، یعنی به ازای هر دو مجموعه‌ی P_1 و P_2 که $P_1 \subset P_2$ است، آنگاه

$$\mu(P_1) \leq \mu(P_2)$$

فرض کنید $\epsilon > 0$ داده شده است، به زیرمجموعه‌ی $Q \subseteq P$ یک ϵ -مجموعه‌ی هسته برای مجموعه P گویند اگر رابطه‌ی زیر برقرار باشد:

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

یکی از مجموعه هسته‌های معروف، مجموعه هسته‌ی مطرح برای تابع اندازه‌گیری عرض نقاط است. به چنین مجموعه هسته‌ای به اختصار ϵ -هسته^{۳۳} گفته می‌شود. تعریف دقیق ϵ -هسته در زیر آمده است:

تعریف ۳-۲ فرض کنید S^{d-1} کره‌ی واحد با مرکز واقع در مبدأ در فضای \mathbb{R}^d باشد. به ازای هر مجموعه P از نقاط در فضای \mathbb{R}^d و هر جهت دلخواه $u \in S^{d-1}$ ، عرض جهت‌دار P در جهت u که با نماد $\omega(u, P)$ تعریف می‌شود، مطابق زیر است:

$$\omega(u, P) = \max_{p \in P} \langle u, p \rangle - \min_{p \in P} \langle u, p \rangle$$

که در آن $\langle \cdot, \cdot \rangle$ همان ضرب داخلی دو بردار است. برای متغیر $\epsilon > 0$ ، زیرمجموعه‌ی $Q \subset P$ یک ϵ -هسته نامیده می‌شود اگر به ازای هر $u \in S^{d-1}$ داشته باشیم:

$$(1 - \epsilon)\omega(u, P) \leq \omega(u, Q)$$

ϵ -هسته یکی از اساسی‌ترین مجموعه هسته‌های مطرح است و برای طیف وسیعی از مسائل قابل استفاده است. الگوریتم‌های زیادی برای محاسبه‌ی ϵ -هسته در حالت ایستا ارائه شده است [۱۷].

^{۳۲}Measure function

^{۳۳} ϵ -Kernel

یکی از روش‌هایی که در تبدیل یک الگوریتم از حالت ایستا به حالت جویبار داده ارائه شده است، استفاده از روش دوبرابری^{۳۴} است. این روش یک روش عام و قابل استفاده برای طیف وسیعی از مسائل است. به عنوان نمونه، دوبرابری را بر روی ϵ -هسته مورد بررسی قرار می‌دهیم. ϵ -هسته، دارای دو خاصیت اساسی زیر است که برای تکنیک دوبرابری قابل استفاده می‌شود.

- اگر P_2 یک ϵ -هسته برای P_1 باشد و P_3 یک δ -هسته برای P_2 باشد، آنگاه P_3 یک $(\epsilon + \delta)$ -هسته برای P_1 است.
- اگر Q_1 یک ϵ -هسته برای P_1 باشد و Q_2 یک ϵ -هسته برای P_2 باشد، آنگاه $Q_1 \cup Q_2$ یک ϵ -هسته برای $P_1 \cup P_2$ است.

ایده‌ی اصلی تکنیک دوبرابری، استفاده از روش مبنای دو است، که در ابتدا از نقطه‌ی اول یک مجموعه هسته ساخته می‌شود. سپس با نقطه‌ی بعدی یک مجموعه هسته از دو نقطه‌ی فعلی ساخته می‌شود. سپس با دو نقطه‌ی بعدی یک مجموعه‌ی هسته از چهار نقطه فعلی ساخته می‌شود. اگر این روند را به صورت توان‌هایی از ۲ ادامه بدهیم، در هر لحظه حداکثر، $\log(n)$ مجموعه هسته داریم که در طول الگوریتم، بعضی از آن‌ها با یکدیگر ادغام می‌گردند و پس از ادغام، یک هسته از کل آن‌ها به عنوان نماینده انتخاب می‌گردد و مجموعه هسته‌ی جدیدی از روی آن ساخته می‌شود. در واقع اگر دوباره بر روی دو مجموعه‌ی ادغام شده هسته حساب نشود، اندازه‌ی هسته به صورت نمایی افزایش می‌یابد که مطلوب نیست. از این رو برای کاهش حافظه‌ی مصرفی از روی دو هسته‌ی ادغام شده، یک هسته‌ی جدید محاسبه می‌گردد. این کار، ضریب تقریب را افزایش می‌دهد ولی باعث می‌شود، حافظه‌ی مصرفی، حداکثر $\log(n)$ برابر حافظه‌ی مصرفی در حالت ایستا می‌شود.

^{۳۴}Doubling

فصل ۳

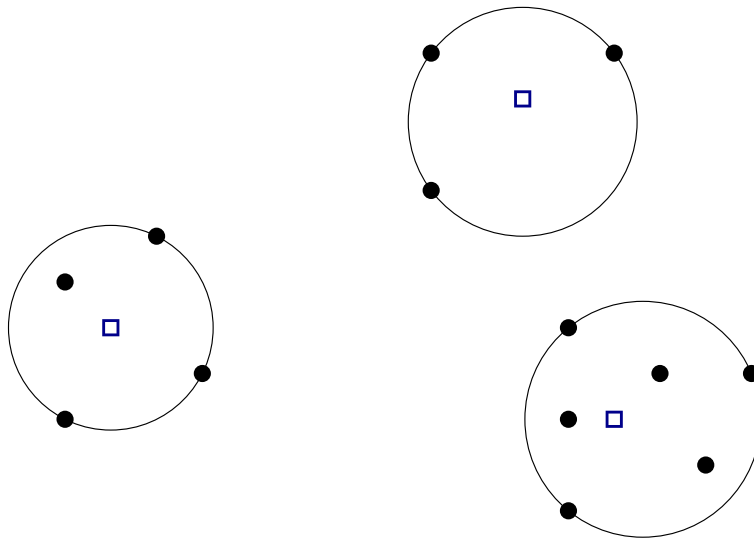
کارهای پیشین

در این فصل کارهای پیشین انجام شده روی مسئله k -مرکز، در سه بخش مورد بررسی قرار می‌گیرد. در بخش اول، مسئله k -مرکز مورد بررسی قرار می‌گیرد. در بخش دوم، حالت جویبار داده‌ی مسئله و مجموعه هسته‌های مطرح برای این مسئله مورد بررسی قرار می‌گیرد و در نهایت، در بخش سوم، مسئله k -مرکز با داده‌های پرت مورد بررسی قرار می‌گیرد.

۳-۱ k -مرکز در حالت ایستا

مسئله k -مرکز به عنوان مسئله‌ای شناخته شده در علوم کامپیوتر مطرح است. این مسئله، در واقع یک مسئله بهینه‌سازی است که سعی در کاهش بیش‌ترین فاصله نقاط از مرکز دسته‌ها را دارد. سختی اصلی این مسئله در انتخاب مرکز دسته‌هاست. زیرا اگر بتوانیم مرکز دسته‌ها را به درستی تشخیص دهیم، کافی است هر نقطه را به دسته‌ای که نزدیک‌ترین مرکز را دارد، تخصیص دهیم. به وضوح چنین تخصیصی بهینه‌ترین تخصیص ممکن است. نمونه‌ای از این تخصیص را در شکل ۳-۱ نشان داده شده است.

در سال ۱۹۷۹، نه تنها اثبات گردید که این مسئله در حالت کلی یک مسئله‌ی ان‌پی-سخت است [۴]، بلکه در سال ۱۹۸۴ ثابت شده است که این مسئله در صفحه‌ی دو بعدی و با متریک اقلیدسی نیز ان‌پی-سخت است [۵]. فراتر از این، ثابت شده است که برای مسئله k -مرکز با متریک دلخواه هیچ الگوریتم تقریبی با ضریب تقریب بهتر از ۲ وجود ندارد.



شکل ۳-۱: نمونه‌ای از تخصیص نقاط به ازای مراکز مربع شکل توخالی.

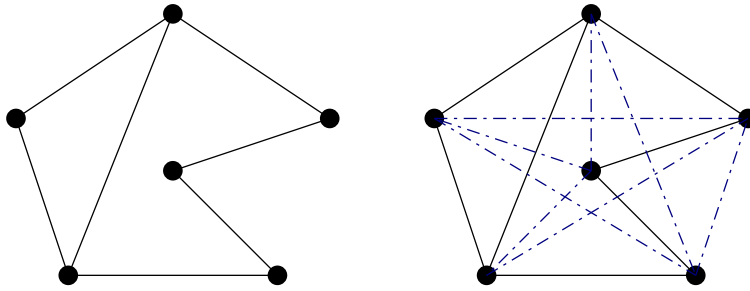
ایده‌ی اصلی این کران پایین، کاهش مسئله‌ی پوشش رأسی، به مسئله‌ی k -مرکز است. برای چنین کاهش‌ی کافی است، از روی گراف اصلی که می‌خواهیم کوچک‌ترین مجموعه‌ی پوشش رأسی را در آن پیدا کنیم، یک گراف کامل بسازیم به طوری که به ازای هر یال در گراف اصلی، یک یال با وزن یک و به ازای هر یال که در گراف اصلی وجود ندارد، یک یال با وزن ۲ قرار دهیم. نمونه‌ای از چنین تبدیلی را در شکل ۳-۲ می‌توانید مشاهده کنید. حال اگر الگوریتمی بتواند مسئله‌ی k -مرکز را با ضریب تقریب بهتر از ۲ حل نماید، آن گاه گراف جدید دارای یک k -مرکز با شعاع کمتر از ۲ است، اگر و تنها اگر گراف اصلی دارای یک پوشش رأسی با اندازه‌ی k باشد. برای متریک L_2 یا فضای اقلیدسی^۱ نیز ثابت شده است برای مسئله‌ی k -مرکز، الگوریتم تقریبی با ضریب تقریب بهتر از $1/822$ وجود ندارد [۱۶].

یکی از اولین الگوریتم‌های تقریبی برای مسئله‌ی k -مرکز به وسیله‌ی گنزالز^۲ ارائه شده است [۱۸]. این الگوریتم یک الگوریتم تقریبی با ضریب ۲ است و در زمان $O(kn)$ قابل اجراست. الگوریتم گنزالز، از نظر روش برخورد با مسئله، یک الگوریتم حریصانه^۳ محسوب می‌شود. برای بیان نحوه‌ی عملکرد الگوریتم گنزالز، نیاز به تعریف فاصله‌ی یک نقطه از یک مجموعه نقطه داریم.

^۱ Euclidean space

^۲ Gonzalez

^۳ Greedy



شکل ۳-۲: نمونه‌ای از تبدیل یک گراف ورودی مسئله پوشش رأسی به یک ورودی مسئله k -مرکز (در گراف سمت چپ، یال‌های سیاه وزن ۱ و یال‌های خط‌چین، وزن ۲ دارند)

الگوریتم ۱ الگوریتم گنزالز

ورودی: V مجموعه نقاط و k تعداد مرکز دسته‌ها

۱: S را برابر مجموعه تهی قرار بده.

۲: عنصر دلخواه از مجموعه نقاط V را به S اضافه کن.

۳: به ازای i بین ۲ تا k :

۴: v را نقطه‌ای از V در نظر بگیرید که بیش‌ترین فاصله را از مجموعه‌ی S دارد.

۵: v را به S اضافه کن.

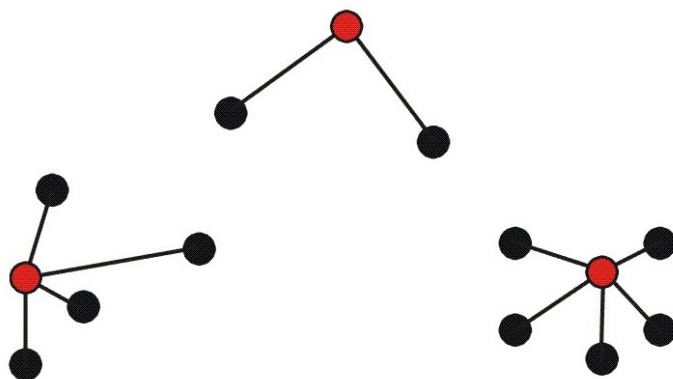
۶: S را برگردان

تعریف ۳-۱ فاصله‌ی نقطه‌ی v از مجموعه‌ای ناتهی از نقاط S را برابر فاصله‌ی نقطه‌ای درون S از v تعریف می‌کنیم، به‌گونه‌ای که از تمام نقاط S به v نزدیک‌تر باشد. در واقع داریم:

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

همان‌طور که در الگوریتم ۱ مشاهده می‌کنید، روش اجرای این الگوریتم به این گونه است که در ابتدا یک نقطه‌ی دلخواه را به عنوان مرکز دسته‌ی اول در نظر می‌گیرد. سپس دورترین نقطه از آن را به عنوان مرکز دسته‌ی دوم در نظر می‌گیرد. در هر مرحله، دورترین نقطه از مرکز مجموعه دسته‌های انتخاب شده را به عنوان مرکز دسته‌ی جدید به مجموعه مراکز دسته‌ها اضافه می‌کند. با اجرای الگوریتم تا k مرحله، مراکز دسته‌ها انتخاب می‌شوند. حال اگر هر نقطه را به نزدیک‌ترین مرکز انتخابی تخصیص دهیم، می‌توان نشان داد که شعاع بزرگ‌ترین دسته، حداکثر دو برابر شعاع بهینه برای مسئله‌ی k -مرکز

است. فدر^۴ و سایرین، زمان اجرای الگوریتم گنزالز را برای هر L_p -متریک به مرتبه‌ی $O(n \log k)$ بهبود بخشیدند. نمونه‌ای از اجرای الگوریتم گنزالز، در شکل ۳-۳ نشان داده شده است.



شکل ۳-۳: نمونه‌ای از حل مسئله‌ی ۳-مرکز با الگوریتم گنزالز

تا به اینجا، تنها بر روی حالت کلی مسئله‌ی k -مرکز صحبت شد و تنها محدودیتی که مورد توجه قرار گرفت، متریک مطرح برای فاصله‌ی نقاط بوده است. در ادامه به بررسی، حالاتی از مسئله‌ی k -مرکز، که k تعداد دسته‌ها یا d ابعاد فضا ثابت باشند می‌پردازیم. آگاروال^۵ و سایرین الگوریتمی دقیق با زمان اجرای $n^{O(k^{1-\frac{1}{d}})}$ برای مسئله k -مرکز در فضای L_p -متریک با ابعاد ثابت d ارائه داده‌اند [۱۹]. قابل توجه است که اگر d ثابت نباشد، مسئله‌ی k -مرکز حتی برای متریک اقلیدسی (L_2 -متریک) با تعداد دسته‌ی ثابت $k \geq 2$ ، ان‌پی-سخت است [۲۰].

علاوه بر حالاتی که ابعاد فضا یا تعداد دسته‌ها ثابت‌اند، مسئله‌ی k -مرکز برای حالتی که مقادیر k و d کوچک هستند، مورد بررسی قرار گرفته‌اند و الگوریتم‌های بهینه‌تری از الگوریتم‌های کلی برای این حالت‌های خاص ارائه شده است. به طور مثال، برای مسئله‌ی ۱-مرکز در فضای اقلیدسی با ابعاد ثابت، الگوریتم خطی با زمان اجرای $O((d+1)n)$ وجود دارد [۲۱]. الگوریتم ارائه شده بر پایه‌ی دو نکته‌ی اساسی بنا شده است. اول اینکه کره‌ی بهینه را می‌توان با حداکثر $d+1$ نقطه‌ی واقع در پوسته‌ی کره‌ی بهینه مشخص نمود و دوم اینکه اگر نقاط ورودی را با ترتیبی تصادفی پیمایش کنیم احتمال اینکه نقطه‌ی پیمایش شده جزء نقاط مرزی باشد از مرتبه‌ی $O(\frac{d}{n})$ است که با توجه به ثابت بودن d این احتمال برای n های بزرگ کوچک محسوب می‌شود و زمان اجرای الگوریتم، به طور میانگین خطی خواهد بود.

^۴Feder

^۵Agarwal

برای متریک اقلیدسی در دو بعد برای مسئله‌ی ۲-مرکز، بهترین الگوریتم را چن^۶ با زمان اجرای $O(n \log^2 n \log^2 \log n)$ و حافظه‌ی $O(n)$ ارائه داده است [۲۲]. برای فضای سه‌بعدی اقلیدسی نیز آگاروال و سایرین، الگوریتمی با متوسط زمان اجرای $O(n^3 \log^4 n)$ ارائه داده است [۲۳].

۳-۲. مرکز در حالت جویبار داده

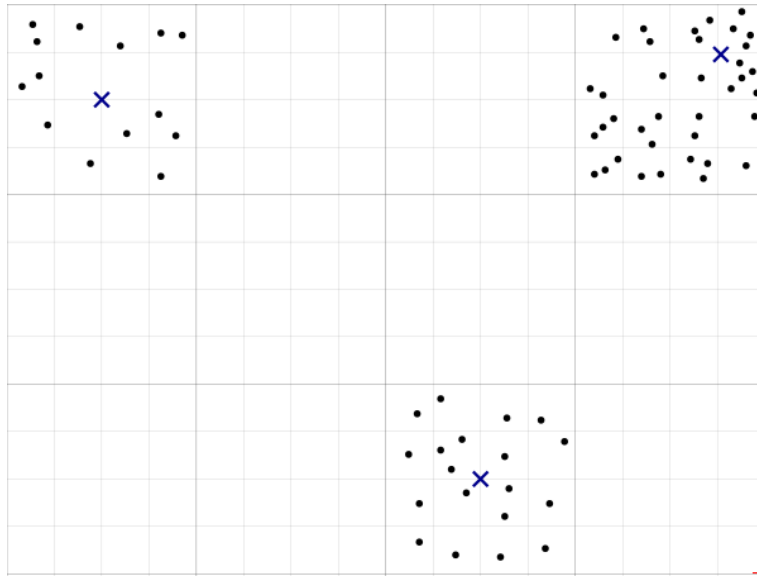
در مدل جویبار داده، مشکل اصلی عدم امکان نگه‌داری تمام داده‌ها در حافظه است و باید سعی شود تنها داده‌هایی را که ممکن است در ادامه مورد نیاز باشند نگه‌داریم. یکی از راه‌های رایج برای این کار نگه‌داری مجموعه‌ای از نقاط (نه لزوماً زیرمجموعه‌ای از نقاط ورودی) به عنوان نماینده‌ی نقاط است به‌طوری‌که جواب مسئله‌ی k -مرکز برای آن‌ها منطبق با جواب مسئله‌ی k -مرکز برای نقاط اصلی باشد (با تقریب قابل قبولی). به چنین مجموعه‌ای مجموعه‌ی هسته‌ی نقاط گفته می‌شود.

بهترین مجموعه هسته‌ای که برای مسئله‌ی k -مرکز در مدل جویبار داده ارائه شده است، روش ارائه‌شده به وسیله‌ی ضربی‌زاده برای نگه‌داری یک ϵ -هسته با حافظه‌ی $O(\frac{k}{\epsilon^d})$ برای L_p -متریک‌ها است [۲۴]. در روش ارائه شده، از چند ایده‌ی ترکیبی استفاده شده است.

در ابتدا، الگوریتم با استفاده از یک الگوریتم تقریبی با ضریب ثابت، یک تقریب از جواب بهینه به دست می‌آورد. به طور مثال با استفاده از الگوریتم گنزالز، یک ۲-تقریب از شعاع بهینه به علاوه‌ی مرکز دسته‌های پیدا شده را به دست می‌آورد. حال با استفاده از طول شعاع الگوریتم تقریبی، حول هر مرکز، یک توری با $O(\frac{1}{\epsilon})$ شبکه‌بندی در هر بعد تشکیل می‌دهد و چون هر نقطه در حداقل یکی از توری‌ها قرار می‌گیرد، می‌توان با حداکثر ϵ تقریب در جواب نهایی نقاط را به نقاط شبکه‌بندی توری گرد نمود. با این کار، دیگر نیازی به نگهداری تمام نقاط ورودی نبوده و تنها نقاط شبکه‌بندی توری نگهداری می‌شود. با این روش می‌توان به یک ϵ -هسته برای مسئله‌ی k -مرکز رسید.

نکته‌ی اساسی برای سازگار سازی روش ارائه‌شده با مدل جویبار داده‌ی تک‌گذره استفاده از روش دوبرابرسازی^۷ رایج در الگوریتم‌های جویبار داده است. نمونه‌ای از اجرای الگوریتم ضربی‌زاده در شکل ۳-۴ نشان داده شده است. برای دیدن اثبات‌ها و توضیح بیشتر در مورد روش ارائه شده می‌توانید به مرجع [۲۴] مراجعه کنید.

^۶Chan^۷Doubling



شکل ۳-۴: نمونه‌ای از شبکه‌بندی الگوریتم ضربی‌زاده (نقاط ضرب در شکل، مراکز به دست آمده از الگوریتم تقریبی است). پس از شبکه‌بندی کافی است برای هر کدام از خانه‌های شبکه‌بندی، تنها یکی را به عنوان نماینده در نظر بگیریم.

از جمله محدودیت‌های الگوریتم ضربی‌زاده، وابستگی اندازه‌ی مجموعه‌ی هسته به ابعاد فضا است. بنابراین هسته‌ی ارائه شده به وسیله‌ی ضربی‌زاده را نمی‌توان برای ابعاد بالا مورد استفاده قرار داد. از طرفی، حساب کردن جواب از روی هسته در زمان چندجمله‌ای بر اساس k و d قابل انجام نیست. با توجه با موارد گفته شده، برای قابل استفاده شدن الگوریتم‌ها برای ابعاد بالا، الگوریتم‌هایی ارائه می‌شود که ضریب تقریب بدتری دارند، اما میزان حافظه‌ی مصرفی یا اندازه‌ی مجموعه هسته‌ی آن‌ها چندجمله‌ای بر اساس d و k و $\log n$ باشد. به چنین الگوریتم‌هایی الگوریتم‌های جویبار داده برای ابعاد بالا گفته می‌شود.

اولین الگوریتم ارائه شده برای ابعاد بالا، الگوریتمی با ضریب تقریب ۸ و حافظه‌ی مصرفی $\mathcal{O}(dk)$ است [۲۵]. پس از آن، گوها^۸، به طور موازی با مک‌کاتن^۹ و سایرین، الگوریتمی با ضریب تقریب $(2+\epsilon)$ با حافظه‌ی مصرفی $\mathcal{O}(\frac{dk}{\epsilon} \log \frac{1}{\epsilon})$ برای مسئله‌ی k -مرکز در هر فضای متریکی ارائه دادند [۷، ۸]. در سال ۲۰۱۴، آهن^{۱۰} و سایرین، الگوریتمی با همین ضریب تقریب و حافظه‌ی $\mathcal{O}((k+3)!2^{k\frac{d}{\epsilon}})$ ارائه داده‌اند که برای k های ثابت، حافظه را از مرتبه‌ی $\mathcal{O}(\log \frac{1}{\epsilon})$ کاهش می‌دهد [۹].

^۸Guha^۹McCutchen^{۱۰}Ahn

تا به اینجا ما به بررسی مسئله k -مرکز در حالت جویبار داده بدون محدودیت خاصی پرداختیم. برای مقادیر کوچک k ، به خصوص ۱ و ۲، مسئله k -مرکز با متریک اقلیدسی مورد بررسی زیادی قرار گرفته است و راه‌حل‌های بهتری نسبت به حالت کلی برای آن‌ها پیشنهاد شده است. به طور مثال، می‌توان یک هسته با اندازه $O(\frac{1}{\epsilon^{d-1}})$ ، با استفاده از نقاط حدی^{۱۱} در تعداد مناسبی جهت، به صورت جویبار داده ساخت.

تعریف ۲-۳ مجموعه‌ی نقاط P را در نظر بگیرید. $Meb(P)$ را برابر تویی با کوچک‌ترین شعاع تعریف می‌کنیم که تمام نقاط P را می‌پوشاند.

همان‌طور که برای الگوریتم ضربی‌زاده مطرح شد، مشکل عمده‌ی مجموعه هسته‌ی ارائه‌شده، وابستگی حافظه‌ی مصرفی آن به d است. در راستای حل این مشکل، ضربی‌زاده و چن [۲۶] برای ابعاد بالا و متریک اقلیدسی، الگوریتمی با ضریب تقریب $1/5$ و حافظه‌ی مصرفی $O(d)$ ارائه دادند. در واقع در الگوریتم آن‌ها، در هر لحظه تنها یک مرکز و یک شعاع نگه داشته می‌شود، که کم‌ترین حافظه‌ی ممکن برای مسئله ۱-مرکز است. همان‌طور که در الگوریتم ۲ مشاهده می‌کنید، نقطه‌ی اول را به عنوان مرکز کره با شعاع صفر در نظر گرفته می‌شود. با فرا رسیدن هر نقطه‌ی جدید، اگر نقطه‌ی مد نظر، داخل کره‌ی فعلی بیفتد که بدون هیچ تغییری ادامه داده و در صورتی که بیرون کره‌ی فعلی بیفتد، آن را با کوچک‌ترین کره‌ای که نقطه‌ی جدید به علاوه‌ی کره‌ی قبلی را به طور کامل می‌پوشاند، جایگزین می‌کند.

به وضوح در هر لحظه کره‌ی ساخته شده تمام نقاطی که تا کنون در جویبار داده آمده است را می‌پوشاند. از طرفی ثابت می‌شود شعاع کره در هر لحظه، حداکثر $1/5$ برابر شعاع کره‌ی بهینه است. نکته‌ی قابل توجه در مورد این الگوریتم، نگهداری کم‌ترین حافظه‌ی ممکن برای مسئله ۱-مرکز است. زیرا تنها یک مرکز و یک شعاع نگه داشته می‌شود. نمونه‌ای از اجرای الگوریتم را بر روی چهار نقطه می‌توان در شکل ۳-۵ دید. برای اثبات کامل‌تر می‌توانید به مرجع [۲۶] مراجعه کنید.

در ادامه، آگاروال و شاراف‌کومار^{۱۲} [۱۰]، الگوریتمی تقریبی با حافظه‌ی مصرفی $O(d)$ ارائه دادند. در الگوریتم ارائه شده، ضریب تقریب، برابر $\frac{1+\sqrt{3}}{4}$ تخمین زده شد، اما با تحلیل دقیق‌تری که چن و

^{۱۱}Extreme points^{۱۲}Sharathkumar

الگوریتم ۲ الگوریتم ضربی زاده و چن

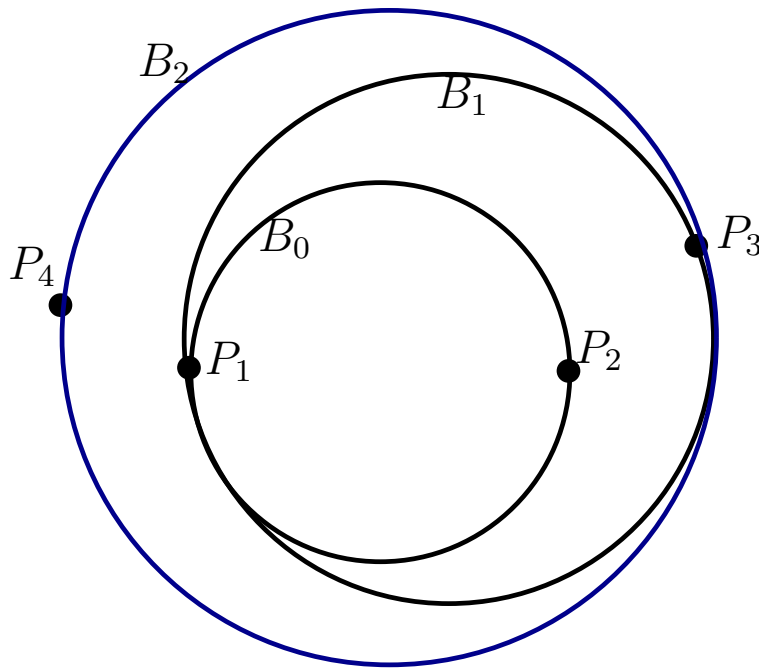
- ۱: B را تویی به مرکز نقطه‌ی اول و شعاع صفر قرار دهید.
- ۲: به ازای هر نقطه‌ی u در جویبار داده:
- ۳: اگر u داخل B قرار می‌گیرد:
- ۴: ادامه بده.
- ۵: در غیر این صورت:
- ۶: B را با کوچک‌ترین توپ ممکن که هردوی B و u را می‌پوشاند، جایگزین کن.
- ۷: مجموعه‌ی S را برگردان.

پفک^{۱۳} [۱۱] بر روی همان الگوریتم انجام دادند، مشخص شد همان الگوریتم دارای ضریب تقریب $1/22$ است.

الگوریتم آگاروال از الگوریتم بایدو^{۱۴} و کلارکسون [۲۷] به عنوان یک زیرالگوریتم استفاده می‌کند. الگوریتم کلارکسون، الگوریتمی کاملاً مشابه الگوریتم گنزالز است و به این گونه عمل می‌کند که در ابتدا یک نقطه دلخواه را به عنوان نقطه‌ی اول انتخاب می‌کند. سپس دورترین نقطه از نقطه‌ی اول را به عنوان دومین نقطه انتخاب می‌کند. از این به بعد در هر مرحله، نقطه‌ای که از توپ Meb نقاط انتخاب شده‌ی قبلی بیش‌ترین فاصله را دارد به عنوان نقطه‌ی جدید انتخاب می‌کند. اگر این الگوریتم را تا $O(1/\epsilon)$ مرحله ادامه بدهیم، به مجموعه‌ای با اندازه‌ی $O(1/\epsilon)$ خواهیم رسید که بایدو و کلارکسون اثبات کرده‌اند که یک ϵ -هسته برای مسئله‌ی ۱-مرکز است.

الگوریتم آگاروال به این گونه عمل می‌کند که اولین نقطه‌ی جویبار داده را به عنوان تنها نقطه‌ی مجموعه‌ی K_1 در نظر می‌گیرد. حال تا وقتی که نقاطی که فرا می‌رسند داخل $(1 + \epsilon)Meb(K_1)$ قرار بگیرند، ادامه می‌دهد. اولین نقطه‌ای که در شرایط ذکر شده صدق نمی‌کند را p_2 بنامید. حال الگوریتم کلارکسون را بر روی $K_1 \cup \{p_2\}$ اجرا کرده و مجموع هسته‌ی به دست آمده را K_2 بنامید. با ادامه‌ی این روند، الگوریتم، دنباله‌ای از مجموعه هسته $\kappa = \{K_1, \dots, K_u\}$ نگه می‌دارد و زمانی که نقطه‌ی p_{u+1} پیدا شود که در هیچ‌کدام از $(1 + \epsilon)Meb(K_j)$ به ازای $1 \leq j \leq u$ نباشد، الگوریتم کلارکسون را برای $\bigcup_{j=1}^u K_j \cup \{p_{u+1}\}$ اجرا نموده و مجموعه هسته‌ی به دست آمده را K_{u+1} می‌نامد. با توجه به نحوه‌ی

^{۱۳}Pathak^{۱۴}Badoiu



شکل ۳-۵: نمونه‌ای از اجرای الگوریتم ضربی‌زاده و چن بر روی چهار نقطه $P_1 \dots P_4$ که به ترتیب اندیس در جویبار داده فرا می‌رسند و دایره‌های B_0, \dots, B_2 دایره‌هایی که الگوریتم به ترتیب نگه می‌دارد. ساخته شدن K_i ها، به راحتی می‌توان نشان داد رابطه‌ی زیر برقرار است:

$$P \subset \bigcup_{i=1}^u (1 + \epsilon) \text{Meb}(K_i)$$

حال در نهایت برای به دست آوردن جواب نهایی کافی است کوچک‌ترین کره‌ای که

$$\bigcup_{i=1}^u (1 + \epsilon) \text{Meb}(K_i)$$

را می‌پوشاند را به عنوان جواب محاسبه کنیم. چن و پفک ثابت کرده‌اند که کره‌ی نهایی دارای شعاع حداکثر $1/22$ برابر شعاع بهینه است. برای مشاهده جزئیات بیشتر، به [۱۰، ۱۱] مراجعه کنید.

آگاروال نه تنها الگوریتمی ارائه داد که در نهایت، ثابت شد حداکثر جوابی با ضریب تقریب $1/22$ برابر جواب بهینه می‌دهد، بلکه نشان داد، که با حافظه‌ی چندجمله‌ای بر اساس $\log n$ و d نمی‌توان الگوریتمی ارائه داد که ضریب تقریب بهتر از $\frac{1+\sqrt{2}}{4}$ داشته باشد.

قضیه ۳-۱ [۱۰] هر الگوریتم تحت مدل جویبار داده که یک α -تقریب برای مسئله‌ی ۱-مرکز برای مجموعه‌ی S شامل n نقطه در فضای \mathbb{R}^d نگه می‌دارد، برای $\alpha \leq \frac{1+\sqrt{2}}{4}(1 - \frac{2}{d^{\frac{1}{d}}})$ با احتمال حداقل $\frac{2}{3}$

نیاز به $\Omega(\min\{n, e^{d^{\frac{1}{3}}}\})$ حافظه مصرف می‌کند.

اثبات. ایده‌ی اصلی اثبات بر اساس محدودیت انتقال اطلاعات بین آلیس و باب^{۱۵} در نظریه انتقال اطلاعات بنا شده است. برای خواندن اثبات این قضیه می‌توانید به مرجع [۱۰] مراجعه کنید.

□

علاوه بر مسئله‌ی ۱-مرکز، مسئله‌ی ۲-مرکز نیز در سال‌های اخیر مورد توجه قرار گرفته است و بهبودهایی نیز برای این مسئله ارائه شده است. کیم و آهن [۱۲] در سال ۲۰۱۴، اولین الگوریتم با ضریب تقریب کم‌تر از ۲ را برای مسئله‌ی ۲-مرکز در فضای اقلیدسی ارائه دادند. این الگوریتم تقریباً پایه‌ی کار این پایان‌نامه برای حالت‌های مختلف است. به همین منظور، تعدادی از لم‌های داخل این الگوریتم که در آینده استفاده می‌شود به اختصار توضیح داده می‌شوند.

لم ۲-۳ [۱۲] فرض کنید B یک کره‌ی واحد با مرکز c در فضای اقلیدسی \mathbb{R}^d باشد. هر پاره‌خط pq به طول حداقل $1/2$ که به طور کامل داخل B قرار دارد، کره‌ی $B'(c, 0/8)$ را قطع می‌کند.

اثبات. صفحه‌ی گذرنده از پاره‌خط و مرکز کره را نظر بگیرید. ادامه اثبات تنها به همین صفحه محدود می‌شود، بنابراین نیاز به در نظر گرفتن ابعاد بزرگ‌تر از ۲ نیست. همان‌طور که در شکل ۳-۶ مشخص شده است، پای عمود از مرکز کره بر پاره‌خط pq را h بنامید. بدون کم شدن از کلیت مسئله فرض کنید $\|hp\| \leq \|hq\|$. بنابراین داریم:

$$0/6 = \frac{1/2}{2} \leq \|hq\|$$

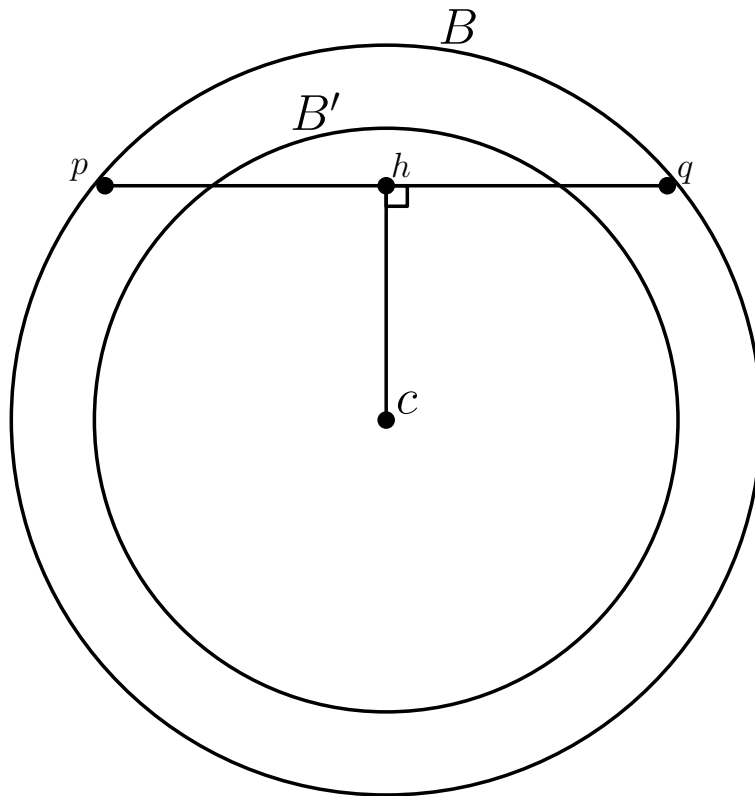
از طرفی چون پاره‌خط pq به طور کامل داخل کره‌ی واحد قرار گرفته است، بنابراین تمام نقاط آن، شامل دو سر آن، از مرکز کره، فاصله‌ی حداکثر ۱ دارند. بنابراین طبق رابطه‌ی فیثاغورث، داریم:

$$\|hc\| = \sqrt{\|qc\|^2 - \|qh\|} \leq \sqrt{1 - 0/6^2} = 0/8$$

بنابراین نقطه‌ی h داخل کره‌ی B' قرار می‌گیرد. از طرفی چون $\|pq\| \leq 1$ است، بنابراین h داخل پاره‌خط قرار دارد و در نتیجه کره‌ی B' با پاره‌خط pq تقاطع دارد.

□

^{۱۵}alice and bob



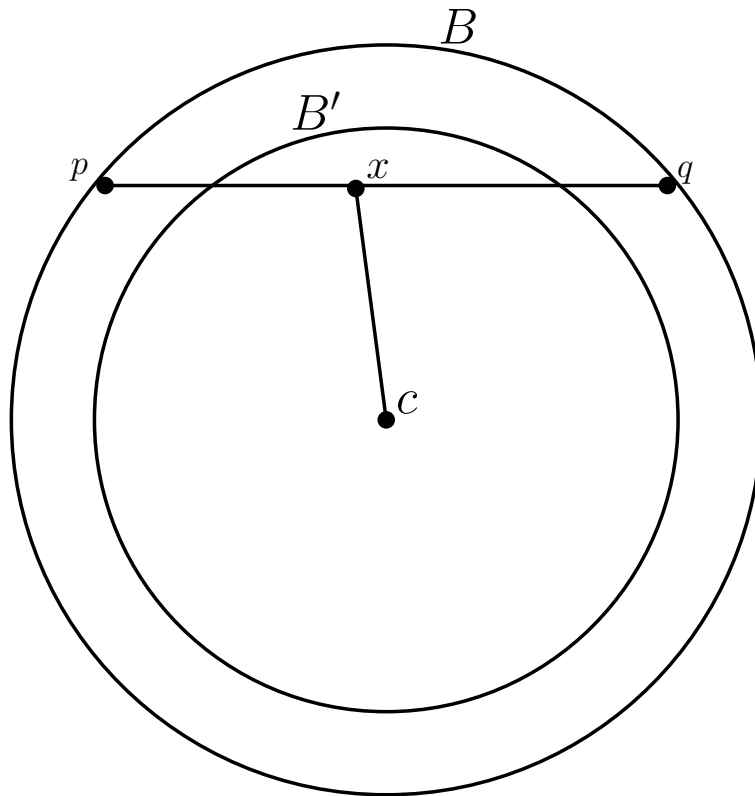
شکل ۳-۶: اثبات لم ۲-۳

لم بالا در واقع نشان می‌دهد اگر در طول الگوریتم بتوانیم دو نقطه‌ی دور نسبت به هم (حداقل $1/2$ برابر شعاع بهینه) از یکی از دو کره‌ی بهینه را بیابیم، پاره‌خط حاصل از مرکز کره‌ی بهینه فاصله‌ی کمی (حداکثر $0/8$ شعاع بهینه) دارد.

لم ۳-۳ [۱۲] فرض کنید B کره‌ای به مرکز c و شعاع واحد در \mathbb{R}^d باشد. پاره‌خط دلخواه pq با طول حداقل $1/2$ که به طور کامل داخل B قرار دارد را در نظر بگیرید. هر نقطه‌ی x از پاره‌خط pq که از دو سر آن حداقل $0/6$ فاصله داشته باشد، داخل کره‌ی $B'(c, 0/8)$ قرار می‌گیرد.

اثبات. اثبات این لم نیز کاملاً مشابه لم ۲-۳ است. بدون کم شدن از کلیت مسئله همان‌طور که در شکل ۷-۳ مشخص شده است، فرض کنید زاویه‌ی $\angle pxc$ بزرگ‌تر مساوی 90° درجه است. در نتیجه داریم:

$$\sqrt{\|px\|^2 + \|xc\|^2} \leq \|pc\| \leq 1$$



شکل ۳-۷: اثبات لم ۳-۳

از طرفی طبق فرض مسئله داریم:

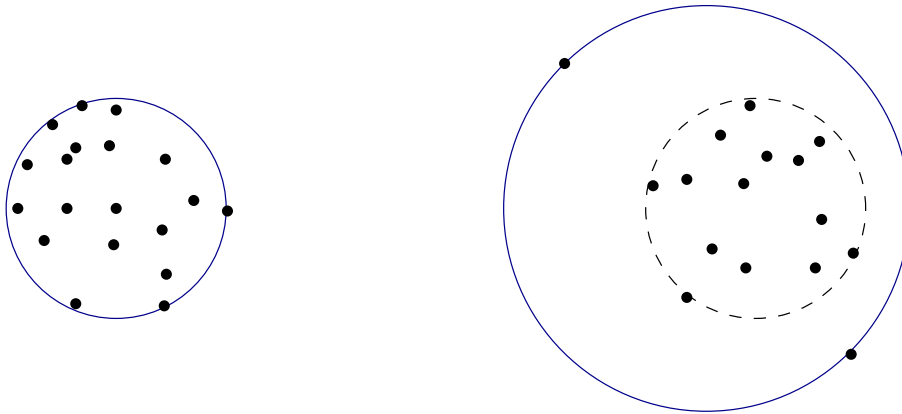
$$0/6 \leq \|px\| \implies \|xc\| \leq \sqrt{1 - 0/6^2} = 0/8$$

□

الگوریتم کیم و آهن، با استفاده از دو لم بالا و تقسیم مسئله به دو حالتی که دو کره‌ی بهینه بیش از ۲ برابر شعاع بهینه یا کم‌تر از ۲ برابر شعاع بهینه فاصله داشته باشند، دو الگوریتم کاملاً جداگانه ارائه می‌دهد که به طور موازی اجرا می‌گردند. اجرای موازی این دو الگوریتم، در هر لحظه دو جواب درست ارائه می‌دهد و کافی است برای جواب نهایی بین دو شعاعی که به عنوان جواب خود می‌دهند، شعاع کم‌تر را به عنوان جواب نهایی الگوریتم ارائه بدهیم.

۳-۳ - مرکز با داده‌های پرت

در دنیای واقعی در میان داده‌ها، داده‌های دارای خطا وجود دارند که اگر امکان تشخیص و حذف آن‌ها در حین جمع‌آوری داده‌ها وجود داشت، شعاع مسئله‌ی k -مرکز به میزان قابل توجهی کاهش پیدا می‌کرد و شهود بسیار بهتری از دسته‌ها ارائه می‌داد. در عمل نیز حذف داده‌هایی که به هیچ دسته‌ای شباهت ندارند، معقول به نظر می‌رسد، زیرا هدف به دست آوردن دسته‌بندی برای کلیت نقاط است و نه دسته‌بندی که تمام نقاط در آن می‌گنجند. همان‌طور که در شکل ۳-۸ می‌بینید، تنها حذف دو نقطه که نسبت به بقیه نقاط داده‌ی اریب حساب می‌شوند، دسته‌بندی بسیار قابل قبول‌تری را نتیجه می‌دهد. با چنین رویکردی، باید تعدادی نقطه که با بقیه نقاط فاصله‌ی زیادی دارند از داده‌های ورودی حذف شده و سپس به عنوان ورودی مسئله‌ی k -مرکز دسته‌های مرتب را از آن استخراج کرد.



شکل ۳-۸: کاهش شعاع مسئله‌ی ۲-مرکز با حذف تنها دو نقطه به عنوان داده‌ی پرت مسئله‌ی k -مرکز با داده‌های پرت، بسیار مشابه مسئله‌ی مکان‌یابی تسهیلات است. در مسئله‌ی مکان‌یابی تسهیلات، هدف مکان‌یابی چند مرکز ارائه‌دهنده‌ی خدمات است که هزینه‌ی مکان‌یابی به علاوه‌ی انتقال تسهیلات از مراکز ارائه‌دهنده به مکان‌های متقاضی کمینه گردد. تعریف رسمی این مسئله در زیر آمده است:

مسئله‌ی ۳-۱ (مکان‌یابی تسهیلات) مجموعه‌ای نقطه به عنوان مکان‌های مجاز برای استقرار تسهیلات داده شده است. هزینه‌ی استقرار تسهیلات در نقطه‌ی i ام را برابر با f_i در نظر بگیرید. مجموعه‌ای از نقاط نیز که متقاضی تسهیلات هستند نیز داده شده است. به ازای هر متقاضی j و محل استقرار تسهیلات i ، $d(i, j)$ برابر هزینه انتقال تسهیلات از محل استقرار به متقاضی است. مجموعه‌ای k عضوی به نام K

انتخاب کنید به طوری که هزینه کلی را کمینه نماید:

$$\sum_{i \in K} f_i + \sum_{\text{all customers}} \min_{i \in K} d(i, j)$$

گونه‌های مختلفی از مسئله مکان‌یابی تسهیلات تعریف شده است. از جمله آن می‌توان به گونه‌های زیر اشاره نمود:

- هر مرکز ارائه‌دهنده حداکثر به تعداد مشخصی از متقاضیان می‌تواند تسهیلات انتقال دهد. در واقع در این روش سعی در مکان‌یابی متوازن تسهیلات است به طوری که متناسب با قدرت ارائه تسهیلات به هر مرکز تقاضا تخصیص یابد.
- هزینه مکان‌یابی مرکز ارائه‌دهنده متناسب با تعداد متقاضیانی که پاسخ می‌دهد است. در این روش، معمولاً هر چه تعداد تقاضاهای یک مرکز بیش‌تر شود هزینه مکان‌یابی یا ساخت آن برای تأمین چنین میزان درخواستی بالاتر می‌رود.
- هر متقاضی میزانی جریمه بابت عدم دریافت تقاضای خود مشخص می‌کند. در این روش، هر متقاضی در صورت عدم دریافت تسهیلات مورد نیاز، میزانی جریمه مطالبه می‌کند و هدف کاهش مجموع هزینه‌ها به علاوه هزینه‌های قبلی است.
- تعدادی از متقاضیان را می‌توان بدون پرداخت جریمه پوشش نداد. در این حالت، امکان چشم‌پوشی از تعدادی از متقاضیان وجود دارد ولی امکان تخطی از محدودیت تعداد آن‌ها وجود ندارد.

اگر به دو گونه آخر توجه بیش‌تری کنید، به شباهتشان به مسئله k -مرکز با داده‌های پرت پی خواهید برد. می‌توان نشان داد که مسئله k -مرکز با z داده‌ی پرت از لحاظ پیچیدگی محاسباتی هم‌ارز مکان‌یابی تسهیلات با امکان عدم پوشش z متقاضی است. همان‌طور که در زیربخش قبلی دیدیم، هیچ الگوریتمی با ضریب تقریب کم‌تر از ۲ برای مسئله k -مرکز در حالت کلی وجود ندارد مگر این‌که $P = NP$ باشد. برای مسئله k -مرکز با داده‌های پرت، اگر تعداد مجاز داده‌های پرت صفر باشد، مسئله به همان مسئله k -مرکز تبدیل می‌گردد.

گونه‌ی مشابهی با مسئله k -مرکز با داده‌های پرت تعریف می‌گردد که در آن تعدادی از نقاط نمی‌توانند به عنوان مرکز انتخاب شوند. به این‌گونه، مسئله k -مرکز با داده‌های پرت و داده‌های ممنوعه

(برای قرارگیری مرکز در آن‌ها) تعریف می‌شود. در مرجع [۳]، ثابت شده است که این مسئله در حالت کلی با ضریب کمتر از ۳ قابل تقریب‌پذیر نیست مگر آن‌که $P = NP$ باشد.

قضیه ۳-۴ فرض کنید نقطه‌هایی دلخواه از یک متریک دلخواه داده شده‌اند. مسئله k -مرکز با داده‌های پرت، که در آن، بعضی از رئوس امکان مرکز شدن ندارند (رئوس ممنوعه)، را نمی‌توان با ضریب تقریبی کمتر از ۳ تقریب زد.

اثبات. ایده‌ی ارائه شده برای این کران پایین، بسیار مشابه با ایده‌ی ارائه شده برای مسئله k -مرکز در فضای اقلیدسی است [۱۶]. در واقع در این راه‌حل، مسئله‌ی بیشینه پوشش مجموعه‌ای، به یک گراف دوبخشی متریک تبدیل می‌گردد که در آن وزن تمام یال‌ها برابر یک است. با استفاده از گراف ساخته شده، نشان داده می‌شود که اگر مسئله k -مرکز با داده‌های پرت و مراکز ممنوعه، الگوریتمی با ضریب تقریب کمتر از ۳ وجود داشته باشد (کوتاه‌ترین مسیر بین دو رأس غیر مجاور در دو بخش مختلف)، آنگاه می‌توان مسئله‌ی بیشینه پوشش مجموعه‌ای را در زمان چندجمله‌ای حل نمود. برای مشاهده‌ی اثبات کامل‌تر می‌توانید به مرجع [۳] مراجعه کنید. \square

الگوریتمی که چریکار^{۱۶} و سایرین در این مقاله ارائه داده‌اند یک الگوریتم ۳-تقریب برای مسئله k -مرکز با داده‌های پرت است. در ابتدا، الگوریتم چریکار، تمام شعاع‌های ممکن را پیدا می‌کند. در حالت مسئله k -مرکز گسسته، کافی است تمام فاصله‌های بین دوه‌دوی نقاط را به عنوان کاندیدا در نظر گرفت که تعدادشان از مرتبه‌ی $O(n^2)$ است و کافی است بروی گزینه‌های به دست آمده، جست‌وجوی دودویی زد. در صورتی که مسئله k -مرکز در حالت پیوسته مد نظر باشد، به ازای هر $d + 1$ نقطه‌ی دلخواه، یک کاندید اضافه می‌گردد که تعدادشان از مرتبه‌ی $O(n^{d+1})$ است. حال اگر شعاع r داشته باشیم که بزرگ‌تر مساوی شعاع بهینه باشد، چریکار یک الگوریتم ساده با شعاع حداکثر $3r$ ارائه می‌دهد که تمام نقاط به غیر از حداکثر z نقطه را می‌پوشاند و اگر الگوریتم چریکار نتوانست با شعاع $3r$ همه‌ی نقاط به جز حداکثر z نقطه را بپوشاند، بنابراین فرض اولیه‌ی ما اشتباه بوده است و r از شعاع بهینه کمتر است.

تعریف ۳-۳ به ازای هر نقطه‌ی $G_i, v_i \in V$ (به طور مشابه) را برابر مجموعه نقاطی در نظر بگیرید که در فاصله‌ی حداکثر r ($3r$ به طور مشابه) از v_i قرار دارند. G_i را توپ به شعاع r و E_i را به عنوان

^{۱۶}Charikar

توپ گسترش یافته به شعاع $3r$ می‌نامیم. وزن هر توپ را برابر تعداد نقاط درون آن در نظر می‌گیریم.

الگوریتم ۳ اولین الگوریتم با ضریب تقریب ۳ برای k – مرکز با z داده‌ی پرت

- ۱: به ازای i از ۱ تا k :
 - ۲: به ازای تمام نقاط، توپ‌ها و توپ‌های گسترش یافته را محاسبه کن.
 - ۳: G_j را توپی در نظر بگیر که بیش‌ترین وزن را دارد (بیش‌ترین تعداد نقاط پوشش داده نشده را می‌پوشاند).
 - ۴: توپ E_j را به عنوان توپ i ام در نظر بگیر.
 - ۵: تمام نقاط داخل E_j را به مجموعه نقاط پوشش داده شده اضافه کن.
 - ۶: اگر همه‌ی نقاط به جز حداکثر z تای آن‌ها پوشانده شده بودند:
 - ۷: r اولیه بزرگ‌تر مساوی r بهینه است. برگردان E_j ‌های انتخاب شده
 - ۸: در غیر این صورت:
 - ۹: r اولیه کوچک‌تر از r بهینه است.
-

الگوریتم ۳، یک الگوریتم حریصانه و ساده است که با مقایسه‌ی عملکرد آن با جواب بهینه می‌توان نشان داد که به درستی عمل می‌کند. برای مشاهده‌ی درستی اثبات، می‌توانید به مرجع [۳] کنید. چریکار در ادامه، با استفاده از برنامه‌ریزی خطی^{۱۷} و گرد کردن^{۱۸} جواب، یک الگوریتم ۳ – تقریب برای حالت گسسته و یک ۴ – تقریب برای حالت پیوسته ارائه می‌دهد. به علت عدم استفاده از روش برنامه‌ریزی خطی، از بیان جزئیات این قسمت صرف نظر می‌کنیم. مک‌کاتن^{۱۹} با استفاده از الگوریتم ارائه شده در بالا، یک الگوریتم جویبار داده ارائه داد که متناسب با اینکه از کدام الگوریتم استفاده کند، همان ضریب تقریب را برای حالت جویبار داده می‌دهد. ایده‌ی اصلی به کار رفته در این تبدیل، پردازش نقاط به صورت دسته‌های $O(kz)$ تایی است و در نظر گرفتن نقاط آزاد به عنوان نقاطی است که هنوز مطمئن نیستیم نقطه‌ی پرت است یا باید پوشانده شوند. این الگوریتم حافظه‌ای از مرتبه‌ی $O(\frac{kz}{\epsilon})$ مصرف می‌کند. برای مشاهده جزئیات بیش‌تر به مرجع [۷] مراجعه کنید.

^{۱۷}Linear Programming

^{۱۸}Rounding

^{۱۹}McCutchen

تا به اینجا مسئله‌ی k - مرکز را در حالت کلی بررسی کردیم. مسئله‌ی ۱ - مرکز با داده‌های پرت به صورت جداگانه به وسیله‌ی ضربایی زاده و موکاپادیه^{۲۰} مورد بررسی قرار گرفته است [۱۳]. در الگوریتم ارائه شده برای حالتی که $z = 1$ است، ضربایی زاده یک الگوریتم با ضریب تقریب $1/48 \leq 1/22 \times \frac{\sqrt{2}+1}{4} \leq$ با حافظه‌ی مصرفی $O(d)$ ارائه داده است. به ازای z های کلی، الگوریتم دیگری با حافظه‌ی مصرفی $O(d^3 z)$ و ضریب تقریب $1/73 \leq 1/22 \times \sqrt{2} \leq$ ارائه شده است.

ایده‌ی اصلی که در این مقاله ارائه شده است، ارائه‌ی یک راهکار کلی برای مسائل جویبار داده است. در این راهکار، یک حافظه‌ی میانگیر^{۲۱} تعریف می‌شود که هر نقطه از جویبار داده به محض ورود به آن اضافه می‌شود. در صورتی که حافظه‌ی میانگیر، پر گردد، یکی از نقاط از حافظه استخراج شده و به الگوریتم زیرین داده می‌شود. الگوریتم زیرین یک الگوریتم جویبار داده برای مسئله‌ی ۱ - مرکز بدون داده‌های پرت است. همان‌طور که در زیر بخش مسئله‌ی k - مرکز در حالت جویبار داده بیان شد، بهترین الگوریتم موجود یک الگوریتم با حافظه‌ی مصرفی $O(d^3 z)$ و ضریب تقریب $1/22$ است.

عامل ثانویه‌ای که در ضریب تقریب نهایی الگوریتم تأثیر به سزایی دارد نحوه‌ی استخراج نقطه از حافظه‌ی میانگیر است. فرض کنید O_x مجموعه نقاطی از P (جویبار داده) باشند که در جواب بهینه به عنوان داده‌ی پرت انتخاب شده‌اند و O مجموعه نقاطی که به اشتباه از حافظه‌ی میانگیر استخراج شدند باشد، اگر داشته باشیم

$$\text{Meb}((P - O_x) \cup O) \leq \beta \text{Meb}(P - O_x)$$

در نتیجه ضریب تقریب نهایی برابر $1/22\beta$ خواهد بود. نکته‌ی قابل توجه در اینجا، تأثیر طول حافظه‌ی میانگیر، در ضریب β است.

در حالت کلی z ، ایده‌ی اصلی برای استخراج یک نقطه از حافظه‌ی میانگیر، نقطه‌ی مرکزی^{۲۲} نقاط داخل حافظه‌ی میانگیر است. در واقع نزدیک‌ترین نقطه به نقطه‌ی مرکزی نقاط داخل حافظه‌ی میانگیر، استخراج می‌گردد و ثابت می‌شود با این شیوه‌ی استخراج $\beta \leq \sqrt{2}$ خواهد بود. برای مشاهده‌ی اثبات و جزئیات بیش‌تر به مرجع [۱۳] مراجعه کنید.

در فصل آتی، ابتدا به پیشرفت‌هایی که برای مسئله‌ی ۱ - مرکز در حالت جویبار داده با داده‌های پرت در این پایان‌نامه ارائه شده است، خواهیم پرداخت. سپس برای مسئله‌ی ۲ - مرکز در حالت جویبار داده

^{۲۰}Mukhopadhyay

^{۲۱}Buffer

^{۲۲}Centerpoint

با داده‌های پرت، اولین کار موجود را ارائه می‌دهیم که بهبود قابل توجهی نسبت به حالت کلی است.

فصل ۴

نتایج جدید

در این فصل نتایج جدید به دست آمده در پایان نامه توضیح داده می شود. این فصل در سه بخش تهیه شده است. بخش اول به بیان مقدمات و نمادگذاری های مورد نیاز برای بخش های بعدی می پردازد. در بخش دوم، راه حل های ارائه شده برای مسئله ی ۱ – مرکز با z داده ی پرت در حالت جویبار داده مورد بررسی قرار می گیرد. این بخش به سه زیربخش تقسیم می شود.

در زیربخش اول، دو الگوریتم جدید ارائه می شود. الگوریتم اول، با مصرف حافظه ی $O(dz^2)$ ، جوابی با ضریب تقریب ۲ ارائه می دهد که حافظه ی مصرفی الگوریتم ضربی زاده و سایرین [۱۳] را در صورتی که ابعاد فضا بیش تر از z باشد بهبود می بخشد. از طرفی الگوریتم ارائه شده بسیار ساده تر از الگوریتم ضربی زاده است و ایده ی مطرح در آن، قابل استفاده برای k های بزرگ تر از ۱ است. الگوریتم دوم یک الگوریتم با ضریب تقریب $\epsilon + 1/8$ و حافظه ی مصرفی $O(\frac{dz^2}{\epsilon})$ است.

در زیر بخش دوم، به بررسی مسئله ی ۱ – مرکز پوشاننده بدون داده ی پرت پرداخته می شود. در این گونه از مسئله ی ۱ – مرکز، هدف علاوه بر پوشش نقاط، پوشش کامل توپ بهینه ی جواب مسئله ی ۱ – مرکز است. برای این مسئله دو الگوریتم متفاوت ارائه می شود. در الگوریتم اول، با استفاده از الگوریتم ارائه شده در زیر بخش قبلی، یک الگوریتم با ضریب تقریب $\epsilon + 1/8$ با حافظه ی مصرفی و زمان به روزرسانی از مرتبه ی $O(\frac{d}{\epsilon})$ ارائه می گردد. در تلاش دوم، یک الگوریتم تقریبی با ضریب تقریب $1/7$ و حافظه ی مصرفی و زمان به روزرسانی از مرتبه ی $O(d)$ ارائه می شود.

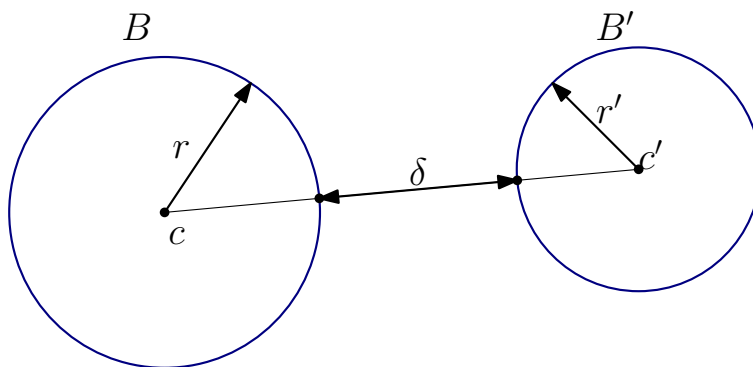
در زیربخش سوم، با استفاده از الگوریتم ارائه شده در زیر بخش قبلی برای الگوریتم ۱ – مرکز پوشاننده

در حالت جویبار داده، الگوریتمی با ضریب تقریب مشابه $1/7$ برای مسئله ۱ - مرکز با z داده‌ی پرت ارائه می‌شود (برای حالتی که z ثابت است).

در بخش سوم، مسئله ۲ - مرکز با z داده‌ی پرت مورد بررسی قرار می‌گیرد. ایده‌ی اصلی این بخش، افراز حالت‌های مسئله به دو حالت مجزا است و ارائه‌ی دو الگوریتم کاملاً متفاوت برای این دو حالت است. هر دو الگوریتم ضریب تقریب $1/8 + \epsilon$ دارند و حافظه‌ی مصرفی در کل برابر است با $O(\frac{dz^4}{\epsilon})$. این اولین الگوریتم ارائه‌شده بعد از الگوریتمی که با ضریب تقریب $4 + \epsilon$ برای k کلی ارائه شده است، می‌باشد که بهبود قابل توجهی محسوب می‌شود.

۴-۱ نمادگذاری‌ها و تعاریف اولیه

در این بخش، تعدادی نمادگذاری که در بخش‌های آتی مورد استفاده قرار می‌گیرند، بیان می‌شود. علاوه بر این، تعدادی از مفاهیم و تعاریف رایج که در بخش‌های آتی به تکرار مورد استفاده قرار می‌گیرند نیز مورد بررسی قرار می‌گیرد.



شکل ۴-۱: تعریف فاصله‌ی دو توپ دلخواه

- در طول متن، برای مشخص کردن یک توپ از نماد $B(c, r)$ استفاده می‌کنیم که c مرکز توپ و r شعاع آن را مشخص می‌کند. هر جا خواستیم به شعاع توپ B ارجاع دهیم از نماد $r(B)$ و هرگاه خواستیم به مرکز توپ B اشاره کنیم از نماد $c(B)$ استفاده می‌کنیم.
- به ازای هر دو نقطه‌ی دلخواه p و q در فضا، فاصله‌ی p و q را با $\|pq\|$ نشان می‌دهیم.
- همان‌طور که در شکل ۴-۱ نشان داده شده است، دو توپ دلخواه $B(c, r)$ و $B'(c', r')$ را در نظر

بگیرید. فاصله‌ی دو توپ B و B' مطابق زیر تعریف می‌شود:

$$\delta(B, B') = \max\{\bullet, \|cc'\| - r - r'\}$$

- دو توپ دلخواه $B(c, r)$ و $B'(c', r')$ را α -جداپذیر^۱ گوئیم اگر داشته باشیم:

$$\delta(B, B') \geq \alpha \cdot \max\{r(B), r(B')\}$$

- زمانی که می‌خواهیم در مورد توپ بهینه‌ی ۱-مرکز برای مجموعه نقاط P صحبت کنیم از نماد $B^*(c^*, r^*)$ یا $\text{MEB}(P)$ استفاده می‌کنیم. برای مسئله‌ی ۲-مرکز نیز از $B_2^*(c_2^*, r_2^*)$ و $B_2^*(c_2^*, r_2^*)$ برای نشان دادن دو دایره‌ی بهینه مسئله‌ی ۲-مرکز برای مجموعه‌ای از نقاط استفاده می‌کنیم.
- به ازای مجموعه‌ی دلخواه P از نقاط فضا، k -دورترین نقطه برای نقطه‌ی دلخواه $p \in P$ ، نقطه‌ای همانند q از مجموعه‌ی P است که در بین تمام نقاط P ، k امین بزرگ‌ترین فاصله از p را دارد.

علاوه بر نمادگذاری‌های بالا، در بخش‌های بعدی، بعضی از تعاریف رایج در هندسه‌ی محاسباتی مورد استفاده قرار می‌گیرد. فرض کنید مجموعه‌ی n عضوی P از نقاط در فضای اقلیدسی \mathbb{R}^d داده شده است. نقطه‌ی $c \in \mathbb{R}^d$ را نقطه‌ی مرکزی^۲ مجموعه‌ی P می‌گویند، اگر هر نیم‌فضای^۳ شامل c ، حداقل شامل $\left\lceil \frac{n}{d+1} \right\rceil$ نقطه از نقاط P باشد. مرجع [۲۸] ثابت کرده است که هر مجموعه‌ی متناهی از نقاط در فضای d -بعدی، دارای یک نقطه‌ی مرکزی است. مشاهده‌ی زیر نتیجه‌ی مستقیم این گزاره است.

مشاهده‌ی ۴-۱ مجموعه‌ی P با $k(d+1)$ نقطه در فضای d -بعدی داده شده است. هر شکل محدب که شامل نقطه‌ی مرکزی P نباشد، حداقل k نقطه از P را نیز نمی‌پوشاند.

در این پایان‌نامه، فرض می‌کنیم ذخیره‌ی هر بعد از یک نقطه حافظه‌ی ثابتی مصرف می‌کند. در نتیجه، ذخیره‌سازی یک نقطه در فضای d -بعدی، $\mathcal{O}(d)$ حافظه مصرف می‌کند و عملیات رایج بر روی نقاط نیز از مرتبه‌ی $\mathcal{O}(d)$ زمان می‌برد.

^۱ α -separable

^۲ Centerpoint

^۳ Half Space

۴-۲ مسئله‌ی ۱- مرکز در حالت جویبار داده

در این بخش به بررسی گونه‌های مختلفی از مسئله‌ی ۱- مرکز در حالت جویبار داده می‌پردازیم. مباحث این بخش، به صورت سه زیربخش دسته‌بندی شده است. در زیربخش اول مسئله‌ی ۱- مرکز با داده‌های پرت، مورد بررسی قرار می‌گیرد. در زیربخش دوم مسئله‌ی ۱- مرکز پوشاننده مورد بررسی قرار می‌گیرد و در نهایت در بخش سوم، با استفاده از نتایج دو بخش قبلی، مسئله‌ی ۱- مرکز با تعداد ثابتی داده‌ی پرت، مورد بررسی قرار می‌گیرد. در هر سه بخش، الگوریتم‌های قبلی از جنبه یا جنبه‌هایی بهبود داده شده‌اند. مهم‌ترین معیارهای مطرح، ضریب تقریب و حافظه‌ی مصرفی است که در هر الگوریتم به دقت محاسبه شده و با کارهای قبلی مقایسه می‌شوند.

۴-۲-۱ مسئله‌ی ۱- مرکز با داده‌های پرت در حالت جویبار داده

در این زیربخش، دو الگوریتم کاملاً متفاوت برای مسئله‌ی ۱- مرکز ارائه می‌شود که نسبت به الگوریتم‌های موجود با ضریب تقریب یکسان ساده‌تر هستند و حافظه‌ی مصرفی کم‌تری دارند. از طرفی دیگر، همان‌طور که در بخش بعدی خواهید دید، ایده‌های مطرح در این الگوریتم‌ها برای ارائه‌ی الگوریتمی تقریبی برای مسئله‌ی ۲- مرکز قابل استفاده هستند.

الگوریتم تقریبی با ضریب تقریب ۲

در این قسمت، یک الگوریتم ساده‌ی جویبار داده با ضریب تقریب ۲ برای مسئله‌ی ۱- مرکز با داده‌ی پرت ارائه می‌شود. در این الگوریتم، از ایده‌ی موازی‌سازی^۴ استفاده می‌شود که به وفور در بخش‌های آتی مورد استفاده قرار می‌گیرد. در الگوریتم ۴ شبه‌کد^۵ الگوریتم ارائه‌شده آمده است. الگوریتم، جویبار داده‌ی P و z تعداد داده‌های پرت را از ورودی دریافت می‌کند. همان‌طور که می‌بینید الگوریتم فرض کرده است که نقطه‌ی اول جویبار داده، داده‌ی پرت نیست. در ادامه نشان خواهیم داد چگونه چنین فرضی را حذف نماییم. در نهایت الگوریتم توپ B را بر می‌گرداند که همه‌ی نقاط P به غیر از حداکثر z نقطه را می‌پوشاند (دقیقاً z دورترین نقطه از نقطه‌ی اول را نمی‌پوشاند).

^۴Parallelization

^۵Pseudocode

الگوریتم ۴ الگوریتم با ضریب تقریب ۲ برای مسئله‌ی ۱ – مرکز با z داده‌ی پرت

۱: c را اولین نقطه از جویبار داده‌ی P قرار بده.

۲: توپ $B(c, 0)$ را در نظر بگیر.

۳: حافظه‌ی میان‌گیر خالی Q را در نظر بگیر.

۴: به ازای هر نقطه‌ی p در جویبار داده‌ی P :

۵: اگر $p \notin B$:

۶: p را به Q اضافه کن.

۷: اگر $|Q| = z + 1$:

۸: q را نزدیک‌ترین نقطه‌ی Q به مرکز c در نظر بگیر.

۹: q را از Q حذف کن.

۱۰: B را با توپ $B(c, \|cq\|)$ جایگزین کن.

۱۱: B را برگردان

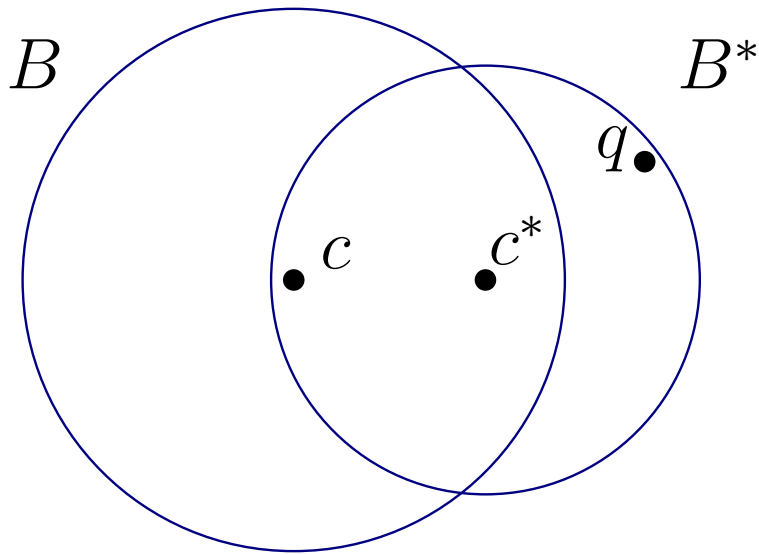
قضیه‌ی ۴-۲ الگوریتم ۴ با فرض این‌که نقطه‌ی اول جویبار داده، داده‌ی پرت نیست یک الگوریتم تقریبی با ضریب تقریب ۲ برای مسئله‌ی ۱ – مرکز با z داده‌ی پرت است.

اثبات. فرض کنید $B^*(c^*, r^*)$ توپ جواب بهینه باشد و c نقطه‌ی دلخواهی از جویبار داده‌ی P است که در جواب بهینه قرار دارد و جزء نقاط پرت نیست. بنابراین c داخل B^* قرار دارد. به ازای هر نقطه‌ی دلخواه $p \in B^*$ داریم:

$$\|cp\| \leq \|cc^*\| + \|c^*p\| \leq 2r^* \quad (۴-۱)$$

از طرفی، از بین $z + 1$ دورترین نقطه از c ، حداقل یک نقطه به نام q وجود دارد که در جواب بهینه داده‌ی پرت نیست و در نتیجه داخل B^* قرار دارد (به شکل ۴-۲ نگاه کنید). با توجه به رابطه‌ی ۴-۱، $\|cq\| \leq 2r^*$ است. از طرفی چون شعاع جواب الگوریتم ۴ به اندازه‌ی فاصله‌ی c از $z + 1$ – دورترین نقطه از c است، بنابراین شعاع جواب الگوریتم نیز کمتر مساوی $2r^*$ است و بنابراین الگوریتم ۴ یک الگوریتم ۲ – تقریب برای مسئله‌ی ۱ – مرکز با z داده‌ی پرت است.

□



شکل ۲-۴: اثبات قضیه‌ی ۲-۴

الگوریتم ۴ به طور ضمنی فرض کرده است که نقطه‌ی اول جویبار داده، در جواب بهینه نقطه‌ی پرت نیست. برای حذف چنین فرضی، $z + 1$ نمونه از الگوریتم ۴ به طور موازی اجرا می‌گردد به طوری که در هر کدام، یکی از $z + 1$ نقطه‌ی اول جویبار داده به عنوان نقطه‌ی اول جویبار داده به الگوریتم ۴ داده می‌شود و بقیه نقاط در ادامه می‌آید. به وضوح، در بین $z + 1$ نقطه‌ی اول، حتماً یک نقطه وجود دارد که در جواب بهینه داده‌ی پرت نیست. بنابراین جواب آن نمونه از الگوریتم، یک ۲-تقریب برای جواب بهینه است و در نتیجه، کوچک‌ترین توپ بین $z + 1$ نمونه‌ی موازی، همواره یک ۲-تقریب برای جواب بهینه است. با توجه به این که پیچیدگی حافظه‌ی الگوریتم ۴ برای یک نمونه از مرتبه‌ی $O(dz)$ است و زمان به‌روزرسانی آن از مرتبه‌ی $O(d + \log z)$ است، نتیجه زیر به دست می‌آید.

قضیه‌ی ۳-۴ برای یک جویبار داده از نقاط در فضای d -بعدی، الگوریتم ۴ یک ۲-تقریب برای مسئله‌ی ۱-مرکز با z داده‌ی پرت با حافظه‌ی مصرفی $O(dz^2)$ و زمان به‌روزرسانی $O(dz + z \log z)$ ارائه می‌دهد. این الگوریتم می‌تواند به هر پرسمان در زمان $O(z)$ پاسخ دهد.

الگوریتم تقریبی با ضریب تقریب ۱/۸

در این قسمت، الگوریتمی با ضریب تقریب ۱/۸ برای مسئله ۱-مرکز با z داده‌ی پرت ارائه می‌دهیم. برای بیان الگوریتم فرض می‌کنیم r' ای داده شده است که در شرایط زیر صدق می‌کند:

$$(2-4) \quad 1/2 r^* \leq r' \leq (1/2 + \frac{2\epsilon}{3}) r^*$$

با فرض داده شدن r' ، الگوریتم ۵، یک توپ با شعاع حداکثر $\frac{3}{4}r'$ ارائه می‌دهد که حداکثر z نقطه از جویبار داده را نمی‌پوشاند. بدون کم شدن از کلیت مسئله، همانند قسمت قبلی فرض کنید که نقطه‌ی اول جویبار داده، جزء نقاط پرت در جواب بهینه نباشد. در نهایت برای حذف چنین فرضی کافی است $1 + z$ نمونه از الگوریتم ارائه شده را به طور موازی اجرا نموده و از بین $1 + z$ توپ جواب، توپ با کوچک‌ترین شعاع را به عنوان جواب نهایی بدهیم. با این تغییر، حافظه‌ی مصرفی، زمان به‌روزرسانی و زمان پاسخ‌گویی به پرسمان همگی در مرتبه‌ی $O(z)$ ضرب می‌شوند. برای ادامه‌ی کار به لم زیر نیاز داریم:

لم ۴-۴ نقطه‌ی q از جویبار داده‌ی P را در نظر بگیرید به‌طوری‌که در توپ بهینه‌ی B^* قرار گرفته است (داده‌ی پرت نیست) و فاصله‌ی آن از p_1 بزرگ‌تر مساوی r' باشد (به شکل شکل ۳-۴ نگاه کنید). نقطه‌ی c را در فاصله‌ی $\frac{1}{4}r'$ از p_1 در راستای پاره‌خط p_1q در نظر بگیرید. ثابت می‌شود توپ $B'(c, \frac{3}{4}r')$ ، توپ B^* و $B(p_1, r')$ را به طور کامل می‌پوشاند. به چنین عملی گسترش توپ $B(p_1, r')$ در راستای نقطه‌ی q گفته می‌شود.

اثبات. طبق فرض لم، طول پاره‌خط pq حداقل $1/2 r^*$ است و پاره‌خط به طور کامل داخل B^* قرار می‌گیرد. بنابراین طبق لم ۳-۳، مرکز توپ B^* که با c^* نشان داده می‌شود، حداکثر $1/8 r^*$ از c فاصله دارد. برای هر نقطه‌ی s که در توپ $B(c, r')$ قرار می‌گیرد، داریم:

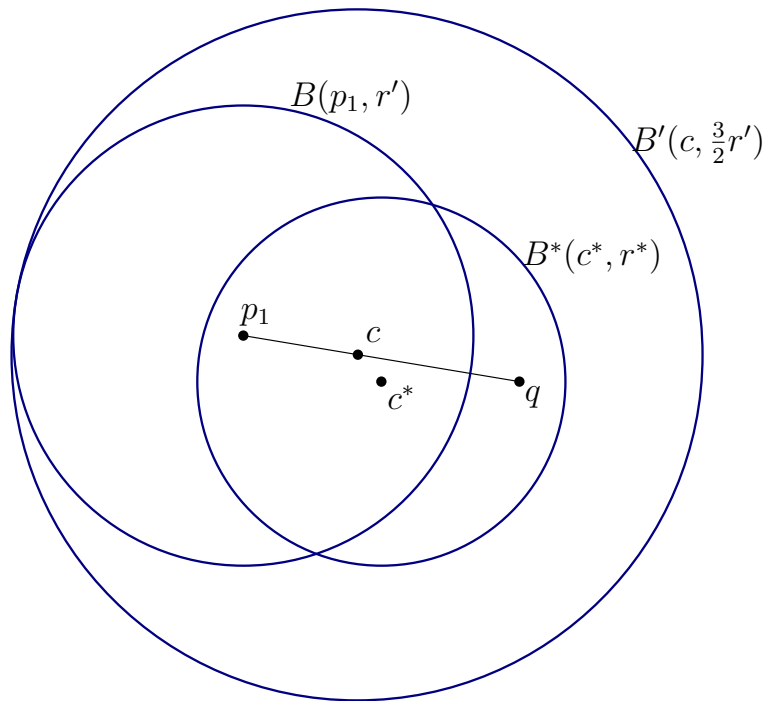
$$\|sc\| \leq \|sp_1\| + \|p_1c\| \leq r' + \frac{1}{4}r' \leq \frac{3}{4}r'$$

در نتیجه $B(p_1, r')$ به طور کامل داخل $B(c, \frac{3}{4}r')$ قرار می‌گیرد. از طرفی برای هر نقطه‌ی s داخل B^* داریم:

$$\|sc\| \leq \|sc^*\| + \|c^*c\| \leq r^* + 1/8 r^* \leq \frac{3}{4}r'$$

و در نتیجه هر نقطه از B^* داخل $B(c, \frac{3}{2}r')$ قرار می‌گیرد و بنابراین $B(c, \frac{3}{2}r')$ به طور کامل B^* را می‌پوشاند.

□



شکل ۳-۴: گسترش توپ $B(c, r')$ در راستای نقطه q

در واقع به عنوان نتیجه مستقیم لم بالا، اگر بتوانیم دو نقطه‌ی غیر پرت با فاصله‌ی بیش‌تر مساوی r' پیدا کنیم، می‌توانیم یک توپ به شعاع $\frac{3}{2}r'$ ارائه دهیم که توپ B^* را به طور کامل می‌پوشاند.

با توجه به لم بالا، با فرض داشتن r' همان‌طور که در الگوریتم ۵ می‌بینید، در ابتدا توپ $B(p_1, r')$ را به عنوان توپ کاندیدا در نظر می‌گیریم. حال نقاطی که خارج این توپ قرار می‌گیرند را داخل یک حافظه‌ی میان‌گیر قرار می‌دهیم. اگر اندازه حافظه‌ی میان‌گیر هیچ‌گاه به $z + 1$ نرسید، بنابراین توپی با شعاع r' پیدا کرده‌ایم که حداکثر z نقطه خارج آن قرار دارد و چون رابطه‌ی ۲-۴ برای r' برقرار است، بنابراین یک جواب با ضریب تقریب $\epsilon + 1/2$ از جواب بهینه به دست آورده‌ایم.

اگر حافظه‌ی میان‌گیر پر شود، حتماً یکی از اعضای آن وجود دارد که جزء داده‌های پرت نبوده (طبق اصل لانه‌کبوتری^۶). بنابراین اگر نسبت به آن نقطه (کافی است تمام گزینه‌ها را امتحان کنیم) توپ اولیه را گسترش دهیم، به توپی می‌رسیم که تمام نقاط قبلی (غیر از نقاط داخل حافظه‌ی میان‌گیر) را پوشانده

^۶Pigeonhole Principle

و مطمئن هستیم کل جواب بهینه را نیز می‌پوشاند.

الگوریتم ۵ الگوریتم با ضریب تقریب $1/8$ برای مسئله ۱ – مرکز با z داده‌ی پرت

۱: فرض کنید r' تقریب برای $1/2r^*$ و z_1 تعداد نقاط پرت قبل از گسترش B داده شده‌اند.

۲: توپ $B(p_1, r')$ را در نظر بگیر.

۳: حافظه‌ی میان‌گیر خالی Q را در نظر بگیر.

۴: به ازای هر نقطه‌ی p در مجموعه‌ی P :

۵: اگر $p \notin B$:

۶: p را به Q اضافه کن.

۷: اگر B هنوز گسترش پیدا نکرده و $|Q| = z_1 + 1$:

۸: توپ B را در راستای p گسترش بده.

۹: به ازای هر $q \in Q$:

۱۰: اگر $q \in B$:

۱۱: q را از Q حذف کن.

۱۲: اگر $|Q| = z + 1$:

۱۳: از برنامه خارج شو.

۱۴: B را برگردان

پس از گسترش هر کدام از گزینه‌ها، کافی است در هر لحظه نگاه‌داریم چند نقطه و کدام نقاط خارج از توپ گسترش‌یافته قرار می‌گیرند. توجه کنید در لحظه‌ی تشکیل توپ گسترش‌یافته، طبق لم بالا، تنها نقاط حافظه‌ی میان‌گیر که تعدادشان $z + 1$ تا است، ممکن است خارج توپ گسترش‌یافته قرار بگیرند و نیازی به نگاه‌داشتن نقاط قبلی نیست.

اگر در ادامه‌ی جویبار داده تعداد نقاط خارج از توپ گسترش‌یافته بیش از z عدد گردد، با پوشش کامل B^* تناقض دارد و در نتیجه با توجه به لم بالا، یا نقطه‌ی q خود جزء داده‌های پرت در جواب بهینه بوده است یا شعاع r' در شرایط گفته شده صدق نمی‌کرده است و در هر صورت گزینه باید حذف گردد. با این حذف گزینه‌ها، در هر لحظه تعدادی گزینه داریم (همواره حداقل یک گزینه وجود دارد، چون وجود دارد حالتی که فرض درستی کرده است) و هر کدام یک جواب با شعاع حداکثر $\frac{3}{4}r'$ ارائه

می‌دهند که اگر از بین آن‌ها توپ با شعاع کمینه را به عنوان جواب نهایی بدهیم، مطمئناً یک جواب با تقریب حداکثر $\epsilon + 1/8$ از جواب بهینه ارائه داده‌ایم.

تا به این جا الگوریتمی ارائه دادیم که با فرض داشتن r' و داده‌ی پرت نبودن p_1 ، با مصرف حافظه‌ی $\mathcal{O}(dz)$ و زمان به‌روزرسانی $\mathcal{O}(dz)$ در هر لحظه می‌تواند یک $\epsilon + 1/8$ - تقریب از جواب بهینه را با مصرف زمان $\mathcal{O}(z)$ ارائه بدهد. توجه کنید که اگر نقاط پرت را نیز بخواهیم گزارش کنیم، نیاز به مصرف حافظه‌ی $\mathcal{O}(dz^2)$ است.

تنها قسمتی که مورد بررسی قرار نگرفته است، نحوه‌ی به‌دست آوردن r' است که در این قسمت به بررسی آن خواهیم پرداخت. در ابتدا برای این‌که ایده‌ی اصلی را درک کنیم فرض کنید که می‌خواهیم یک الگوریتم دو گذره برای این مسئله ارائه دهیم، به‌طوری‌که در گذر اول، r' محاسبه می‌شود و در گذر دوم، با استفاده از r' به‌دست آمده و الگوریتم ۵، یک $\epsilon + 1/8$ - تقریب ارائه می‌گردد. در ادامه نشان داده می‌شود، چگونه می‌توان این دو گذر را هم‌زمان اجرا نمود. برای پیدا کردن r' کافی است که با استفاده از یک الگوریتم α - تقریب (به طور مثال الگوریتم ۲ - تقریب ارائه شده در قسمت قبلی)، یک r به‌دست آوریم. طبق الگوریتم استفاده شده داریم:

$$r^* \leq r \leq \alpha r^*$$

حال اگر بازه‌ی $[0, 1/2r]$ را به

$$m = \left\lceil \frac{3}{\epsilon} \times \frac{1}{2} \times \alpha \times \frac{1}{\epsilon} \right\rceil$$

قسمت مساوی تقسیم کنیم، طول هر قسمت برابر است با:

$$\frac{1/2r}{m} = \frac{2}{3} \times \frac{r}{\alpha} \times \epsilon \leq \frac{2\epsilon}{3} r^*$$

از طرفی چون $1/2r^* \leq 1/2r$ است، بنابراین یکی از این بازه‌ها $1/2r^*$ را شامل می‌شود و انتهای آن بازه با توجه به طول بازه‌ها، کاندیدای مناسبی برای r' است. بنابراین کافی است پس از پیدا کردن یک α - تقریب برای مسئله‌ی ۱ - مرکز با z داده‌ی پرت، به ازای سرهای تمام بازه‌ها، الگوریتم ۵ را اجرا کنیم و از بین گزینه‌هایی که از اجرای هر نمونه باقی می‌مانند کوچک‌ترین توپ را به عنوان جواب نهایی بدهیم. با توجه به این‌که برای سر یکی از بازه‌ها، r در رابطه‌ی ۴-۲ صدق می‌کند، در نتیجه یکی از گزینه‌ها یک $\epsilon + 1/8$ - تقریب برای جواب بهینه است و در نتیجه توپ با شعاع کمینه نیز همین ضریب تقریب را تضمین می‌کند.

تنها قسمتی باقی مانده که نیازمند بررسی بیش‌تری است، چگونگی تک‌گذره کردن الگوریتم است. همان‌طور که گفته شد، از الگوریتم ۴ که الگوریتمی ۲- تقریب است، برای پیدا کردن r' استفاده می‌کنیم. فرض کنید r_i برابر شعاع الگوریتم ۲- تقریب برای جویبار داده تا i آمین عنصر جویبار داده باشد. به وضوح دنباله‌ی r_i یک دنباله صعودی است. به ازای هر k, i را عدد صحیحی در نظر بگیرید که رابطه‌ی زیر برقرار باشد:

$$2^{k-1} \leq r_i \leq 2^k$$

$l_i = 2^k$ قرار دهید. به وضوح طبق رابطه‌ی گفته شده، $l_i \leq 2r_i$ است و در نتیجه l_i یک ۴- تقریب برای مسئله‌ی ۱- مرکز با z داده‌ی پرت است.

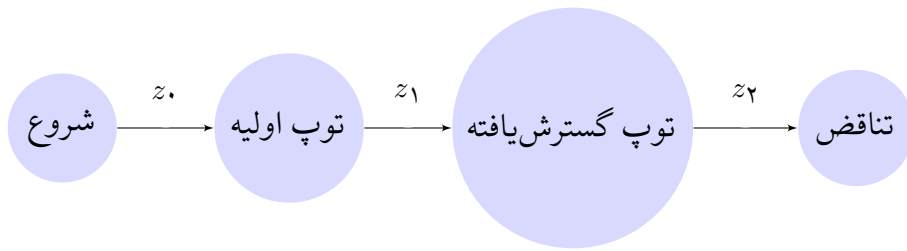
حال کافی است بازه‌ی $[0, 1/2 l_i]$ را به $m = \left\lceil \frac{1/2 l_i}{\epsilon} \right\rceil$ قسمت تقسیم کنیم. با این تقسیم‌بندی، طول هر بازه، $t_i = \frac{1/2 l_i}{m}$ می‌شود و مجموعه سر بازه‌ها برابر

$$R_i = \{j \times t_i \mid 1 \leq j \leq m\}$$

می‌گردد. طبق توضیحات قسمت قبل، سر یکی از بازه‌ها کاندیدای مناسبی برای r' است.

حال کافی است که در هر لحظه m نمونه از الگوریتم ۵ را به ازای هر $r \in R_i$ به صورت موازی اجرا نماییم. به ازای اضافه شدن نقطه‌ی p_i ، اگر $l_i = l_{i-1}$ باشد، $R_i = R_{i-1}$ است و در نتیجه بدون هیچ تغییری کافی است p_i را به تمام نمونه‌های موازی اضافه کنیم. در حالتی که $l_{i-1} < l_i$ باشد، مجموعه‌ی R_i را می‌توان به دو زیرمجموعه تقسیم نمود. اعضای $r \in R_i$ که کمتر مساوی $1/2 l_{i-1}$ هستند و در نتیجه داخل R_{i-1} نیز قرار دارند (چون $\frac{t_i}{t_{i-1}}$ همواره توان صحیحی از دو است). برای چنین اعضای، کافی است، نمونه معادل آن را در R_{i-1} بدون هیچ تغییری پیدا کرد و نقطه‌ی p_i را به آن اضافه کرد.

در حالت دوم، اگر $r \in R_i$ باشد و در R_{i-1} نباشد، در نتیجه $1/2 l_{i-1} \leq r \leq l_{i-1}$ است. با توجه با نحوه‌ی عملکرد الگوریتم ۴، می‌دانیم در هر لحظه z نقطه‌ای به عنوان داده‌ی پرت در نظر گرفته می‌شود که فاصله‌ی بزرگ‌تر مساوی l_i دارند و بقیه نقاطی که تا کنون آمده‌اند فاصله‌ای کمتر مساوی l_i دارند. بنابراین در بین تمام نقاط جویبار داده تا کنون، حداکثر z نقطه‌ی ذخیره شده در حافظه‌ی میان‌گیر الگوریتم ۴ خارج توپ $B(p_1, r)$ می‌افتند. بنابراین اگر بخواهیم به ازای این r جدید الگوریتم ۵ را بر روی نقاط جویبار داده تا کنون اجرا نماییم، کافی است الگوریتم را به ازای نقاط داخل حافظه‌ی میان‌گیر اجرا نموده و مطمئن هستیم که بقیه‌ی نقاط به علت قرارگیری داخل $B(p_1, r)$ ، تأثیری در روند اجرای الگوریتم نخواهند داشت.



شکل ۴-۴: نحوه اجرای الگوریتم ۵

با جمع‌بندی روند توضیح داده شده، ساخت یک نمونه‌ی جدید از الگوریتم ۵ معادل اضافه کردن حداکثر z نقطه‌ی موجود در حافظه‌ی میان‌گیر الگوریتم ۴ به نمونه‌ی جدید از الگوریتم که از مرتبه‌ی $O(dz^2)$ زمان می‌برد. در هر مرحله هم حداکثر m نمونه‌ی جدید ساخته می‌شود، بنابراین زمان به‌روزرسانی نمونه‌ها در هر مرحله حداکثر $O(\frac{dz^2}{\epsilon})$ است. از طرفی در هر لحظه m نمونه موازی از الگوریتم ۵ در حال اجراست. بنابراین، زمان به‌روزرسانی نهایی الگوریتم برابر $O(\frac{dz^2}{\epsilon})$ است و حافظه‌ی مصرفی نیز متناسب با m نمونه‌ی موازی از الگوریتم ۵ برابر $O(\frac{dz}{\epsilon})$ است. اگر بخواهیم نقاط پرت را نیز گزارش کنیم حافظه‌ی مصرفی برابر $O(\frac{dz^2}{\epsilon})$ خواهد بود. در نهایت با دخیل کردن امکان پرت بودن نقطه‌ی اول جویبار داده، به قضیه‌ی زیر می‌رسیم:

قضیه ۴-۵ الگوریتم ۵ با مصرف حافظه‌ی $O(\frac{dz^2}{\epsilon})$ و زمان به‌روزرسانی $O(\frac{dz^3}{\epsilon})$ ، در هر لحظه با صرف زمان اجرای $O(\frac{z}{\epsilon})$ جوابی با ضریب تقریب $\epsilon + 1/8$ ارائه می‌دهد.

اگر بخواهیم الگوریتم گفته شده را جمع‌بندی کنیم، الگوریتم همان‌طور که در شکل ۴-۴ نشان داده شده است، ابتدا z_0 (تعداد نقاطی که از اول جویبار داده که در جواب بهینه داده‌ی پرت است) نقطه‌ی اول را به عنوان داده‌ی پرت در نظر می‌گیرد و سپس $z_0 + 1$ آمین نقطه‌ی جویبار داده را به عنوان نقطه‌ی اول و مرکز توپ B به شعاع r' در نظر می‌گیرد. همان‌طور که در الگوریتم ۵ گفته شده است، سپس z_1 نقطه‌ی اولی از ادامه‌ی جویبار داده که بیرون این توپ قرار می‌گیرند را به عنوان داده‌ی پرت در نظر گرفته و به ازای $z_1 + 1$ آمین نقطه‌ی خارج B ، آن را در همان راستا گسترش می‌دهد. سپس z_2 نقطه‌ی دیگر از ادامه‌ی جویبار داده که خارج B گسترش یافته قرار می‌گیرند را نیز به عنوان داده‌ی پرت در نظر می‌گیرد، حال اگر نقطه‌ی دیگری در جویبار داده وجود داشته باشد که خارج B بیفتد با توجه به اینکه $\sum_{i=1}^2 z_i = z$ است، تعداد نقاط پرت از z بیش‌تر شده و نشان می‌دهد یکی از فرض‌های اولیه اشتباه بوده و در نتیجه، گزینه حذف می‌گردد.

۴-۲-۲ مسئله‌ی ۱- مرکز پوشاننده در حالت جویبار داده

در این زیر بخش به بررسی مسئله‌ی نسبتاً جدیدی می‌پردازیم. در ابتدا به تعریف دقیق مسئله توجه کنید:

تعریف ۴-۱ مجموعه نقاط P داده شده است. به توپ B یک α -تقریب برای مسئله‌ی ۱- مرکز پوشاننده گویند اگر نه تنها تمام نقاط P را بپوشاند، بلکه توپ بهینه‌ی ۱- مرکز این نقاط که با B^* نشان داده می‌شود را نیز به طور کامل بپوشاند و شعاع آن، حداکثر α برابر شعاع توپ بهینه باشد.

اگر این مسئله را در حالت جویبار داده در نظر بگیریم، هدف نگهداری مجموعه هسته‌ای است که بتوان با استفاده از آن، در هر لحظه یک α -تقریب از مسئله‌ی ۱- مرکز در حالت پوشاننده ارائه داد. در این زیر بخش، برای این مسئله دو الگوریتم ارائه می‌شود. در الگوریتم اول، الگوریتم $\epsilon + 1/8$ -تقریب ارائه شده برای مسئله‌ی ۱- مرکز با z داده‌ی پرت را به گونه‌ای تغییر می‌دهیم که الگوریتمی با ضریب تقریب $\epsilon + 1/8$ برای مسئله‌ی ۱- مرکز پوشاننده در حالت جویبار داده ارائه دهد. در الگوریتم دوم، با استفاده از الگوریتمی دلخواه برای مسئله‌ی ۱- مرکز در حالت جویبار داده به عنوان جعبه‌ی سیاه^۷، یک الگوریتم برای مسئله‌ی ۱- مرکز پوشاننده در حالت جویبار داده ارائه می‌شود.

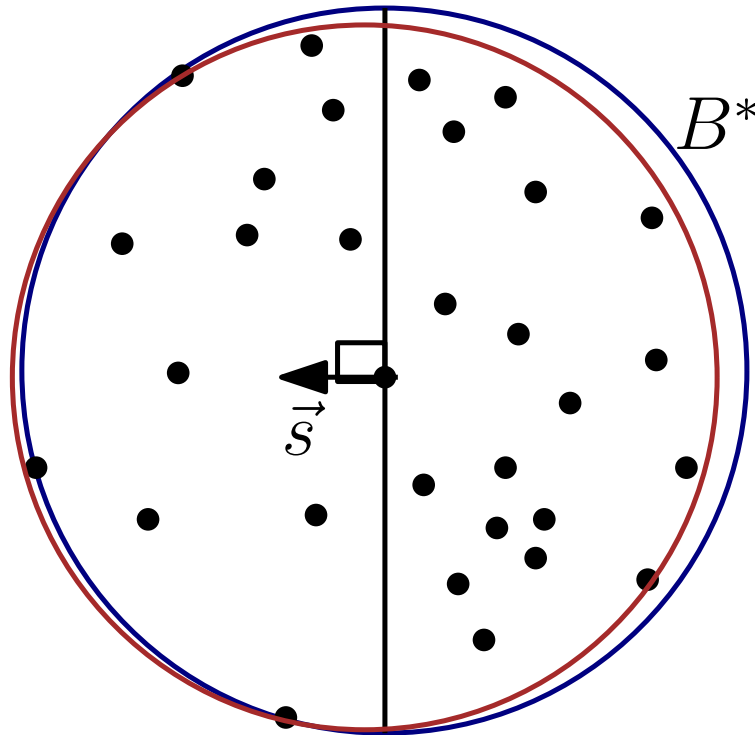
الگوریتم $\epsilon + 1/8$ -تقریب برای مسئله‌ی ۱- مرکز پوشاننده در حالت جویبار داده

اگر الگوریتم ۵، را با $z = 0$ بر روی جویبار داده اجرا کنیم، با مصرف حافظه و زمان به‌روزرسانی از مرتبه‌ی $O(\frac{d}{\epsilon})$ می‌توان یک جواب $1/8$ -تقریب از جواب بهینه برای مسئله‌ی ۱- مرکز به دست آورد. توجه کنید برای حالتی که $z = 0$ است، تنها لازم است m نمونه از الگوریتم ۵ به طور موازی اجرا نمود که هر نمونه از مرتبه‌ی $O(d)$ حافظه مصرف می‌کند. تنها تفاوتی که با الگوریتم قبلی در این استفاده‌ی جدید وجود دارد این است که، در زمان به دست آوردن جواب نهایی، از بین m گزینه، آن گزینه‌ای را به عنوان جواب نهایی می‌دهیم که کم‌ترین r' را دارد (نه گزینه‌ای که کم‌ترین شعاع را داشته باشد). با توجه به این که توپ با r' کم‌تری در گزینه‌ها نیست، بنابراین مطمئن هستیم که برای کم‌ترین مقدار r' بین گزینه‌ها (که با r'_m نشان می‌دهیم)، داریم:

$$r'_m \leq (1/2 + \frac{2\epsilon}{3})r^*$$

^۷Black Box

زیرا مطمئن هستیم r' ای که در رابطه‌ی ۴-۲ صدق می‌کند، در بین گزینه‌ها قرار دارد. حال اگر تویی که با استفاده از r'_m ساخته شده باشد، شعاعی برابر با $\frac{3}{4}r'_m$ داشته است، مطمئن هستیم که B^* را به طور کامل می‌پوشاند و از طرفی شعاعش حداکثر $1/8$ برابر r^* است. اما اگر توپ گسترش نیافته باشند، یک توپ داریم که تمام نقاط را می‌پوشاند و شعاعش حداکثر $1/2$ برابر شعاع بهینه است. برای ادامه، نیاز به دو لم داریم:



شکل ۴-۵: اثبات لم ۴-۶

لم ۴-۶ فرض کنید مجموعه نقاط P در فضای \mathbb{R}^d داده شده‌اند. B^* را تویی با شعاع کمینه در نظر بگیرید که تمام نقاط را می‌پوشاند. آنگاه پوسته‌ی هر نیم‌کره از B^* شامل حداقل یک نقطه از P است.

اثبات. از برهان خلف استفاده می‌کنیم. همان‌طور که در شکل ۴-۵ نشان داده شده است، فرض کنید نیم‌کره‌ای از B^* وجود داشته باشد که بر روی پوسته‌ی آن هیچ نقطه‌ای از P قرار ندارد. بردار عمود بر صفحه‌ای که کره‌ی B^* را به دو نیم کره تقسیم می‌کند را \vec{s} بنامید (رو به خارج نیم‌کره). حال کافی است توپ B^* را به اندازه‌ی بسیار کمی (کم‌تر از فاصله‌ی نقاط توپ و نقاط P) در جهت \vec{s} حرکت دهیم. با این حرکت، هیچ نقطه‌ای بر روی نیم‌کره قرار نمی‌گیرد و پوسته‌ی نیم‌کره‌ی مقابل نیز کاملاً خالی می‌شود. بنابراین می‌توان شعاع توپ را کاهش داد و به توپ قهوه‌ای رنگ که شعاع کم‌تری نسبت به B^* دارد

رسید به طوری که کماکان تمام نقاط را بپوشاند که با بهینه بودن B^* تناقض دارد. بنابراین فرض اولیه مبنی بر وجود نیم‌کره‌ای با پوسته‌ی خالی اشتباه بوده است.

□

لم ۴-۶ در مراجع بسیاری از جمله مرجع [۱۰] استفاده شده است.

لم ۴-۷ فرض کنید مجموعه نقاط P در فضای \mathbb{R}^d داده شده‌اند. $B^*(c^*, r^*)$ را توپی با شعاع کمینه در نظر بگیرید که تمام نقاط را می‌پوشاند. همچنین توپ دلخواه $B(c, r)$ با شعاع αr^* در نظر بگیرید که تمام نقاط P را می‌پوشاند. اگر شعاع توپ B را $\sqrt{2}$ برابر کنیم، کل توپ B^* را نیز می‌پوشاند.

اثبات. همان‌طور که در شکل ۴-۶ نشان داده شده است، هر دوی B و B^* کلیه‌ی نقاط را می‌پوشانند. صفحه‌ی عمود بر پاره‌خط واصل c^*c را نظر بگیرید. مرز تقاطع صفحه‌ی رسم شده با توپ B^* یک دایره به نام D تشکیل می‌دهد. همان‌طور که در شکل پیداست، تمام نقاط این دایره از c به یک فاصله‌اند (به علت عمود بودن c^*c بر صفحه‌ی رسم شده)، بنابراین دایره‌ی D یا به طور کامل داخل B قرار می‌گیرد یا بیرون آن. نقطه‌ی دلخواه q را بر روی دایره‌ی D در نظر بگیرید. چون این نقطه بر روی پوسته‌ی B^* قرار دارد، بنابراین داریم:

$$\|c^*q\| = r^*$$

ادعا می‌کنیم، دایره‌ی D به طور کامل داخل B قرار می‌گیرد، زیرا در صورتی که خارج B قرار بگیرد، پوسته‌ی نیم‌کره‌ی حاصل از تقاطع این صفحه که c در آن قرار نمی‌گیرد، به طور کامل خارج از B قرار می‌گیرد. زیرا به ازای هر نقطه دلخواه t از پوسته‌ی این نیم‌کره، زاویه‌ی $\angle cc^*t$ زاویه‌ای باز است و با توجه به قائم بودن زاویه‌ی $\angle cc^*q$ داریم:

$$\alpha r^* \leq |cq| \leq |ct|$$

از طرفی طبق لم ۴-۶، حداقل یک نقطه به نام s از P بر روی پوسته‌ی این نیم‌کره قرار دارد، که می‌توان نتیجه گرفت s به وسیله‌ی B پوشانده نمی‌شود. بنابراین با فرض لم مبنی بر پوشش تمام نقاط تناقض دارد. پس دایره‌ی D به طور کامل داخل B قرار می‌گیرد.

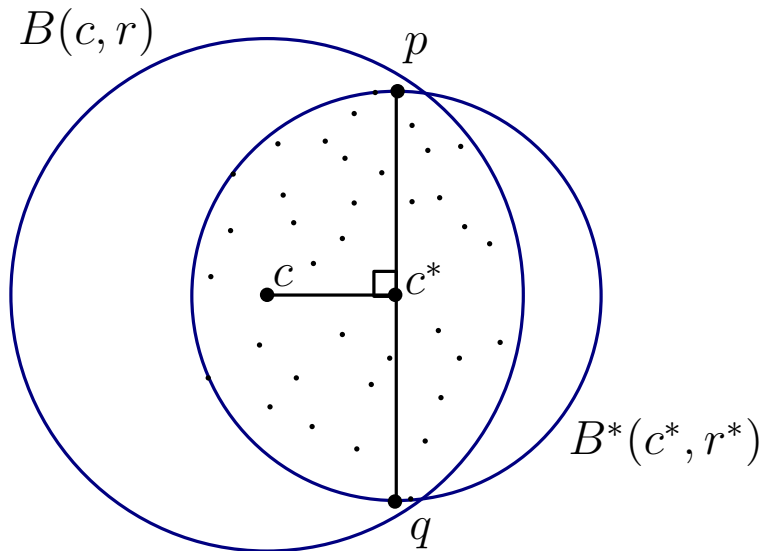
از طرفی دیگر، چون $q \in B$ است بنابراین داریم:

$$\|cq\| \leq r = \alpha r^*$$

و چون زاویه $\angle cc^*q$ قائم است، طبق رابطه فیثاغورث داریم:

$$\|cc^*\| = \sqrt{\|cq\|^2 - \|c^*q\|^2} \leq \sqrt{\alpha^2 - 1}r^*$$

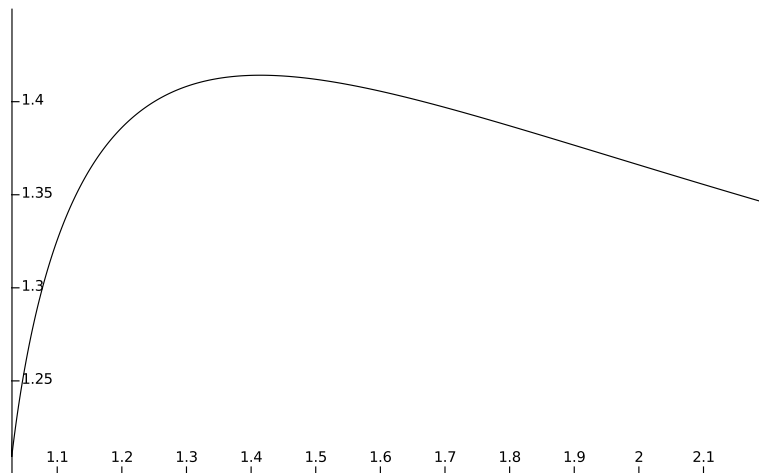
حال با توجه به این که $\alpha \leq 1$ است، اگر شعاع دایره B را $\sqrt{\alpha^2 - 1}r^*$ افزایش دهیم، دایره B^* را به طور کامل می پوشاند. در واقع برای افزایش شعاع به این میزان کافی است، شعاع را $\frac{1+\sqrt{\alpha^2-1}}{\alpha}$ برابر کنیم. اگر مطابق شکل ۷-۴ نمودار این تابع را رسم کنیم، می بینیم که حداکثر تابع در نقطه $\sqrt{2}$ و برابر $\sqrt{2}$ خواهد بود. توجه کنید که اگر $\alpha \leq \sqrt{2}$ باشد، تابع در بازه $[1, \alpha]$ مقداری کمتر از $\sqrt{2}$ به خود می گیرد و همان طور که در نمودار پیداست، چون تابع در این بازه صعودی است، مقدار بیشینه در خود α به دست می آید.



شکل ۷-۴: اثبات لم ۷-۴

□

با توجه به لم ۷-۴، برای حالتی که توپ گسترش پیدا نکرده است، اگر شعاعش را $\sqrt{2}$ برابر کنیم، تضمین می کند که دایره بهینه را به طور کامل می پوشاند. با توجه به این که در این حالت شعاع توپ کمتر مساوی $1/2 r^*$ است، با $\sqrt{2}$ برابر کردن شعاعش به تویی با شعاعی حداکثر $1/8 \times \sqrt{2} \leq 1/2$ برابر r^* دست خواهیم یافت. بنابراین در هر دو حالت، تویی را بر می گردانیم که توپ بهینه را به طور کامل می پوشاند و شعاع آن حداکثر $1/8 + \epsilon$ برابر جواب بهینه است.



شکل ۴-۷: نمودار تابع $\frac{1+\sqrt{\alpha^2-1}}{\alpha}$

الگوریتم ۱/۷ - تقریب برای مسئله ۱ - مرکز پوشاننده در حالت جویبار داده

در این قسمت با استفاده از لم ۴-۷، الگوریتمی با ضریب تقریب ۱/۷ برای مسئله ۱ - مرکز پوشاننده در حالت جویبار داده ارائه می‌دهیم. ایده‌ی اصلی این الگوریتم، استفاده از یک الگوریتم α -تقریب برای مسئله ۱ - مرکز در حالت جویبار داده است و از افزایش شعاع با استفاده از لم ۴-۷، در زمان پاسخ‌گویی به پرسمان برای پوشش کامل توپ بهینه استفاده می‌شود. همان‌طور که در فصل کارهای پیشین ذکر شده است، بهترین الگوریتم موجود برای مسئله ۱ - مرکز در حالت جویبار داده، الگوریتم ارائه شده به وسیله‌ی آگاروال با حافظه‌ی مصرفی و زمان به‌روزرسانی $O(d)$ و ضریب تقریب ۱/۲۲ است. حال اگر جواب این الگوریتم را بخواهیم افزایش بدهیم، طبق لر ۴-۷، باید شعاع آن را $\frac{1+\sqrt{1/22^2-1}}{1/22}$ برابر کنیم، که تویی با شعاع حداکثر ۱/۷ برابر شعاع توپ بهینه ارائه می‌دهد. زمان اجرا و حافظه‌ی مصرفی این الگوریتم همانند الگوریتم آگاروال، $O(d)$ است.

۴-۲-۳ مسئله ۱ - مرکز با داده‌های پرت در حالت جویبار داده با تعداد داده‌های پرت ثابت

در این زیر بخش، با استفاده از الگوریتم‌هایی که در بخش قبل برای مسئله ۱ - مرکز پوشاننده در حالت جویبار داده ارائه شده است، الگوریتم جدیدی برای مسئله ۱ - مرکز با داده‌های پرت در حالت جویبار داده با تعداد داده‌های پرت ثابت ارائه می‌دهیم. در ابتدا لمی را ثابت می‌کنیم که نقش اساسی در اجرای

الگوریتم دارد.

لم ۴-۸ مجموعه‌ی P از نقاط در فضای \mathbb{R}^d در نظر بگیرید. حداکثر $(d+1)^z$ حالت برای انتخاب نقاط پرت این مجموعه وجود دارد. در واقع حداکثر $(d+1)^z$ زیرمجموعه‌ی z عضوی از P وجود دارد که دایره با شعاع کمینه‌ای که سایر نقاط P را می‌پوشاند، هیچ‌کدام از اعضای زیرمجموعه‌ی انتخابی را نپوشاند.

اثبات. از استقراء برای اثبات لم استفاده می‌کنیم.

- پایه: حکم برای $z = 0$ برقرار است، زیرا در این حالت تنها یک حالت برای انتخاب مجموعه‌ی داده‌های پرت وجود دارد. (مجموعه‌ی \emptyset)
- فرض: فرض کنید که به ازای مجموعه‌ی دلخواه P در \mathbb{R}^d ، و $z = k - 1$ حداکثر $(d+1)^z$ حالت برای انتخاب زیرمجموعه داده‌های پرت وجود داشته باشد.
- حکم: ثابت می‌کنیم به ازای هر مجموعه‌ی دلخواه P در \mathbb{R}^d و $z = k$ حداکثر $(d+1)^z$ حالت برای انتخاب زیرمجموعه داده‌های پرت وجود دارد. توپ به شعاع کمینه B^* که تمام نقاط P را می‌پوشاند را در نظر بگیرید. مجموعه‌ی $S \subset P$ را که بر روی پوسته‌ی B^* قرار دارند را در نظر بگیرید. از بین اعضای S می‌توان زیرمجموعه‌ی حداکثر $d+1$ عضوی S' را انتخاب کرد به طوری که توپ محیطی نقاط S' همان توپ B^* باشد (در فضای d -بعدی، هر توپ را با حداکثر $d+1$ نقطه روی پوسته‌ی آن می‌توان مشخص کرد).
- زیرمجموعه‌ی دلخواه O از داده‌های پرت را در نظر بگیرید. اگر $O \cap S' = \emptyset$ باشد، آنگاه کوچک‌ترین توپی که $P - O$ را می‌پوشاند همان B^* است، زیرا S' به طور کامل داخل $P - O$ قرار می‌گیرد. بنابراین فرض تهی بودن اشتراک O و S' غلط است و در نتیجه حداقل یکی از اعضای S' داخل O است. این عضو $d+1$ حالت برای انتخاب دارد. اگر این نقطه را از O و P حذف کنیم، مسئله به تعداد حالت‌های انتخاب داده‌های پرت با اندازه‌ی $k-1$ از مجموعه‌ی جدید تبدیل می‌شود که طبق فرض استقراء $(d+1)^{z-1}$ حالت دارد. در نتیجه در کل تعداد حالات انتخاب O حداکثر برابر $(d+1)^z$ است.

□

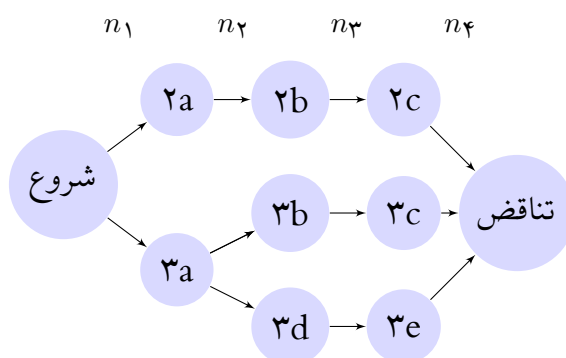
الگوریتم به گونه‌ای عمل می‌کند که تعدادی حالت مختلف را به طور موازی دنبال می‌کند. در ابتدا، قبل از ورود اولین نقطه، تنها یک حالت داریم (حالتی که مجموعه نقاط غیر پرت و پرت هردو تهی هستند). به ازای ورود نقطه‌ی جدید p از جویبار داده، به ازای هر کدام از حالت‌ها، آن را با دو حالت جایگزین می‌کنیم. اولین حالت، حالتی است که نقطه‌ی جدید را به مجموعه نقاط پرت حالت اولیه اضافه می‌کند و دومین حالت، حالتی است که آن را به مجموعه نقاط غیر پرت حالت اولیه اضافه می‌کند. توجه کنید که مجموعه نقاط پرت، یک حافظه با اندازه‌ی حداکثر z و مجموعه نقاط غیر پرت، همان اجرای الگوریتم ۱ – مرکز پوشاننده در حالت جویبار داده است.

یک گزینه در صورتی که تعداد نقاط پرتش از z بیش‌تر شود یا اینکه یکی از نقاط پرتش داخل توپ پوشاننده‌ی نقاط غیر پرت قرار بگیرد، حذف می‌گردد. با توجه به این‌که برای نقاط غیر پرت، از الگوریتمی استفاده می‌کنیم که توپ بهینه را به طور کامل می‌پوشاند، تعداد حالات نقاط پرت نیز در این حالت، حداکثر $(d+1)^z$ حالت است. بنابراین الگوریتمی ارائه دادیم که مسئله‌ی ۱ – مرکز با z داده‌ی پرت در حالت جویبار داده را با ضریب تقریب $1/7$ (حاصل استفاده از الگوریتم ارائه شده در قسمت قبل برای مسئله‌ی ۱ – مرکز پوشاننده در حالت جویبار داده) حل می‌نماید. حافظه‌ی مصرفی و زمان به‌روزرسانی الگوریتم ارائه شده از مرتبه‌ی $O(d \times (d+1)^z)$ است.

۴-۳ مسئله‌ی ۲ – مرکز با داده‌های پرت در حالت جویبار داده

در این بخش، یک الگوریتم $\epsilon + 1/8$ – تقریب برای مسئله‌ی ۲ – مرکز با داده‌های پرت در حالت جویبار داده ارائه می‌شود. در تمام الگوریتم‌های ارائه شده، فرض شده است که نقطه‌ی اول جویبار داده p_1 ، داده‌ای غیر پرت است. این فرض را می‌توان همانند روشی که برای حذف این محدودیت در بخش دوم همین فصل ارائه دادیم، با در نظر گرفتن $1 + z$ نمونه از الگوریتم ارائه شده برطرف نمود.

همان‌طور که در بخش نمادگذاری‌ها ذکر شده بود، $B_1^*(c_1^*, r^*)$ و $B_P^*(c_P^*, r^*)$ را توپ‌های جواب بهینه برای مسئله‌ی ۲ – مرکز با z داده‌ی پرت برای جویبار داده‌ی P در نظر بگیرید. شعاع توپ‌های بهینه را با r^* و فاصله‌ی دو توپ بهینه را با δ^* نشان می‌دهیم. برای این‌که بتوانیم به نتیجه مطلوب برسیم، مسئله را به دو حالت تقسیم می‌کنیم. در زیر بخش اول به بررسی حالت $\delta^* \leq \alpha r^*$ و در زیر بخش دوم به بررسی حالت $\delta^* > \alpha r^*$ می‌پردازیم، که در آن α یک عدد ثابت است که در طول تحلیل نشان داده می‌شود ۱۶ یک انتخاب مناسب برای α است.



شکل ۴-۸: حالت‌های گراف انتقال در الگوریتم ارائه‌شده به وسیله‌ی کیم و آهن [۱۲]

$$\delta^* \leq \alpha r^* \quad \text{حالت ۱-۳-۴}$$

ایده‌ی اصلی این بخش، تغییر الگوریتم ارائه شده به وسیله‌ی کیم^۸ و آهن^۹ [۱۲] که در اصل برای نگه‌داری مجموعه‌ی هسته برای مسئله‌ی ۲-مرکز در حالت جویبارداده با ضریب تقریب $\epsilon + 1/8$ ارائه شده است، حاصل می‌گردد. تغییرات اعمال شده نیز بسیار شبیه عملکرد الگوریتم $\epsilon + 1/8$ -تقریب برای مسئله‌ی ۱-مرکز با داده‌ی پرت است. بنابراین، در این قسمت، برای جلوگیری از تکرار، در ابتدا گام‌های اصلی الگوریتم کیم و آهن را بیان کرده و سپس تغییراتی که در الگوریتم جدید مورد نیاز است را ذکر می‌کنیم.

همان‌طور که در شکل ۴-۸ نشان داده شده است، الگوریتم کیم و آهن، دارای ۱۰ حالت مختلف است. متناسب با نقاطی که تا کنون در جویبار داده آمده‌اند، الگوریتم در یکی از حالت‌های بالا قرار می‌گیرد. در هر کدام یک از حالت‌ها، الگوریتم، دو توپ به عنوان نماینده‌ی جواب در این حالت در نظر می‌گیرد. انتقال بین حالت‌ها، تنها زمانی رخ می‌دهد که نقطه‌ای در جویبار داده وارد شود که در هیچ کدام از دو توپ کاندیدا قرار نگیرد.

الگوریتم کیم و آهن، از گره‌ی شروع، شروع می‌کند و با رسیدن نقاط جدید از جویبار داده در طول گراف مطابق با یال‌ها جابه‌جا می‌گردد. در بعضی از حالت‌ها، بیش از یک حالت برای حالت بعدی وجود دارد (گره‌ی معادل آن حالت، درجه‌ی خروجی بیش از یک دارد) و الگوریتم هیچ اطلاعات قبلی ندارد که کدام یک حالت را به عنوان حالت بعدی انتخاب کند. اما اگر بیش‌تر دقت کنید، تنها ۳ مسیر

^۸Kim

^۹Ahn

از گرهی شروع به گرهی انتهایی وجود دارد. بنابراین کافی است، در ابتدا سه نمونه‌ی موازی از الگوریتم به طور موازی اجرا کنیم که هر کدام به صورت قطعی^{۱۰} مسیر تعیین شده را دنبال می‌کند و در هر لحظه مطمئن هستیم که حداقل یکی از سه مسیر، مسیر درستی است.

در دو زیر بخش بعدی، تغییراتی که در الگوریتم آهن و کیم برای مسئله‌ی ۲ – مرکز با داده‌های پرت ارائه دادیم را بیان می‌کنیم. در بخش اول، تغییرات اصلی در الگوریتم برای تشخیص داده‌های پرت را ارائه می‌دهیم و در زیر بخش دوم، به نحوه‌ی پیدا کردن r' مورد نیاز الگوریتم اصلی می‌پردازیم (تعریف r' کاملاً مشابه r' استفاده شده در الگوریتم $\epsilon + 1/8$ – تقریب برای مسئله‌ی ۱ – مرکز با داده‌های پرت در همین پایان‌نامه است).

الگوریتم اصلی

در این بخش تغییراتی که بر روی الگوریتم کیم و آهن ارائه دادیم را بیان می‌کنیم. الگوریتم ارائه شده، کاملاً مشابه الگوریتم ارائه شده برای مسئله‌ی ۱ – مرکز با داده‌های پرت است که در همین فصل مورد بررسی قرار گرفت. تغییر اصلی الگوریتم جدید، بر روی قسمت انتقال بین حالات اعمال شده است.

در طول اجرای الگوریتم، هر نقطه اگر داخل دو توپ کاندیدا قرار بگیرد باعث تغییر حالت الگوریتم نمی‌گردد. بنابراین حذف چنین نقاطی در روند اجرای الگوریتم تغییری ایجاد نمی‌کند. توجه کنید اگر یک نقطه در داخل دو توپ کاندیدا یک حالت قرار بگیرد، در دو توپ حالت‌هایی که از این حالت قابل رسیدن هستند نیز قرار می‌گیرد، زیرا زمانی که از یک حالت به حالت جدید می‌رویم، کاندیداها به گونه‌ای تغییر می‌کنند که کاندیداها ی قبلی را به طور کامل می‌پوشانند.

بنابراین تنها وجود نقاطی در جویبار داده مهم هستند که خارج توپ‌های کاندیدا قرار می‌گیرند. با توجه به این‌که این نقاط تنها باعث افزایش شعاع توپ‌های کاندیدا می‌گردند و وجودشان در روند الگوریتم تاثیر دارد، بنابراین تنها گزینه‌های مطرح برای نقاط پرت محسوب می‌شوند.

از طرفی چون در هر حالت، تعداد نقاط پرت غیر مشخص است، مجبور هستیم تمام حالت‌های ممکن برای تعداد نقاط پرت را در نظر بگیریم. چون گراف تغییر حالات^{۱۱}، یک گراف جهت‌دار بدون دور با عمق ۴ است، کافی است به ازای هر عمق گراف، تعداد نقاط پرت (n_i) مشخص کنیم و برای

^{۱۰} Deterministic

^{۱۱} Transition Graph

در نظر گرفتن تمام حالات ممکن، تمام ۴-تایی‌های صحیح نامنفی (n_1, \dots, n_4) که $\sum_{i=1}^4 n_i = z$ را در نظر بگیریم. به راحتی می‌توان نشان داد که تعداد چنین ۴-تایی‌هایی از مرتبه $O(z^3)$ است.

الگوریتم ۶ مسئله‌ی ۲-مرکز در حالت نزدیک

ورودی: مجموعه نقاط P ، عدد ثابت r' در بازه‌ی $[1/2r^*, (1/2r^* + \frac{2}{3}\epsilon)r^*]$ و z تعداد نقاط پرت

۱: مجموعه جواب S را برابر \emptyset قرار بده.

۲: به ازای هر ۴-تایی (n_1, \dots, n_4) که $\sum_{i=1}^4 n_i = z$:

۳: به ازای هر $\pi \in \{1, 2, 3\}$: انتخاب یکی از سه مسیر مختلف

۴: به ازای هر i از بین $\{1, \dots, 4\}$:

۵: $counter_i$ را برابر صفر قرار بده.

۶: B_1 را $B(p_1, r')$ قرار بده.

۷: B_2 را مجموعه‌ی \emptyset قرار بده.

۸: متغیر j را برابر ۱ قرار بده. Δ متغیر j عمق حالت را مشخص می‌کند.

۹: به ازای هر $p \in P$:

۱۰: اگر $p \notin B_1 \cup B_2$:

۱۱: $counter_j$ را یک عدد افزایش بده.

۱۲: اگر $counter_j > n_j$:

۱۳: j را یک عدد افزایش بده. Δ در مسیر π به حالت بعدی برو

۱۴: توپ‌های کاندیدا (B_1, B_2) را با توپ‌های کاندیدای حاصل از انتقال حالت

مطابق مسیر π در الگوریتم کیم و آهن جایگزین کن.

۱۵: اگر $j \leq 4$:

۱۶: شعاع توپ با شعاع بیشینه از بین دو توپ B_1 و B_2 را به مجموعه S اضافه کن.

۱۷: کم‌ترین شعاع داخل مجموعه‌ی S را برگردان

شبه‌کد الگوریتم ارائه شده در الگوریتم ۶ نشان داده شده است. به ازای تمام حالت‌های ممکن برای n_1 تا n_4 و هر مسیر مجاز از بین سه مسیر موجود بین گره‌ی شروع تا گره‌ی پایان، الگوریتم یک جواب کاندیدا (B_1, B_2) برای پوشش نقاط غیر پرت نگه می‌دارد. متغیر j ، برای هر حالت، عمق آن حالت را

مشخص می‌کند. چهار شمارنده نیز برای شمارش تعداد نقاطی که در هر عمق به عنوان داده‌ی پرت در نظر گرفته شده‌اند استفاده می‌شود.

الگوریتم ابتدا با دو کاندیدای $B_1 = B(p_1, r')$ و $B_2 = \emptyset$ شروع می‌کند که معادل حالت شروع الگوریتم کیم و آهن است. پس از ورود هر نقطه‌ی p از جویبار داده، در ابتدا بررسی می‌شود که نقطه‌ی مورد نظر در توپ‌های کاندیدا قرار می‌گیرند یا نه. اگر قرار بگیرند به سراغ نقطه‌ی بعدی می‌رویم. در غیر این صورت، اگر تعداد نقاط پرت در این عمق (فرض کنید در عمق j ام هستیم) به n_j نرسیده باشد، نقطه‌ی p را به عنوان داده‌ی پرت در نظر می‌گیریم و به سراغ نقطه‌ی بعدی می‌رویم. در غیر این صورت، مطابق مسیر انتخاب شده، به حالت بعدی می‌رویم و توپ‌های کاندیدای (B_1, B_2) را مطابق با الگوریتم کیم و آهن، به‌روزرسانی می‌کنیم. توجه کنید که برای انتقال حالت مطابق الگوریتم کیم و آهن، علاوه بر نقطه‌ی p ، مسیر حرکت π نیز داده می‌شود تا به طور قطعی، حالت بعدی مشخص شود.

زمانی که تمام نقاط P پردازش شدند، اگر هنوز به گره‌ی پایان وارد نشده‌ایم، توپ‌های کاندیدا را به عنوان یک جواب به مجموعه جواب اضافه می‌کنیم. در غیر این صورت مطابق عملکرد الگوریتم کیم و آهن این حالت، از بین حالات موجود حذف می‌شود. در نهایت از بین تمام جواب‌های ممکن، بهترین جواب را با کم‌ترین شعاع به عنوان جواب نهایی می‌دهیم. همان‌طور کیم و آهن [۱۲] ثابت کرده‌اند، جوابی که با این روش محاسبه می‌شود دارای شعاعی حداکثر $\frac{3}{4}r^*$ است، با فرض اینکه $\delta^* \leq \alpha r^*$ باشد (اثبات بیان شده برای $\alpha = 2$ بیان شده است، اما می‌توان نشان داد که به ازای هر α ثابتی این اثبات صادق است). بنابراین با فرض برقرار بودن رابطه‌ی ۴-۲، قضیه زیر برقرار است:

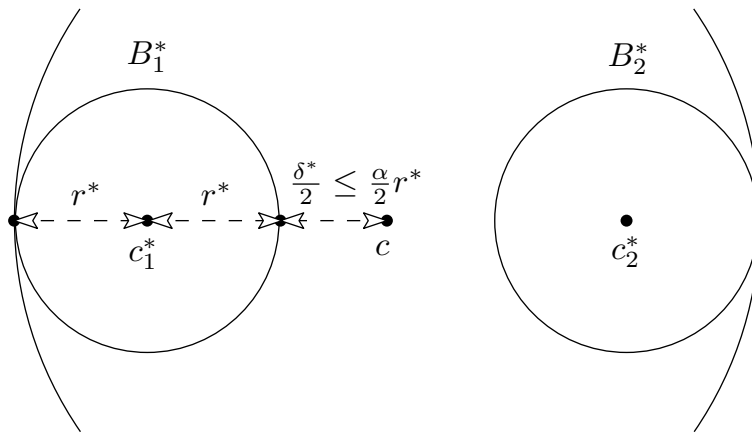
قضیه‌ی ۴-۹ به ازای $r^* \leq (1/2 + \frac{\epsilon}{4})r^* \leq 1/2 r^* \leq r'$ داده شده و با فرض $\delta^* \leq \alpha r^*$ ، الگوریتم ۶ یک جواب $\epsilon + 1/8 -$ تقریب برای جواب بهینه‌ی مسئله‌ی ۲- مرکز با z داده‌ی پرت ارائه می‌دهد. حافظه‌ی مصرفی این الگوریتم، زمان به‌روزرسانی و پاسخ‌گویی به پرسمان آن، از مرتبه‌ی $O(dz^3)$ است (با فرض پرت نبودن داده‌ی اول).

توجه کنید اگر بخواهیم علاوه بر دو توپ جواب، مجموعه نقاط پرت را بدهیم، مجبور هستیم برای هر حالت، یک حافظه‌ی میان‌گیر شامل نقاط پرت در آن حالت در نظر بگیریم، در نتیجه حافظه‌ی مصرفی در این حالت، از مرتبه‌ی $O(dz^4)$ می‌گردد. در زیر بخش بعدی، نحوه‌ی پیدا کردن r' مناسب را مورد بررسی قرار می‌دهیم.

پیدا کردن r'

در این زیر بخش، نشان می‌دهیم که چگونه r' مناسبی را پیدا کنیم که در رابطه‌ی ۴-۲ صدق کند. لم زیر ایده‌ی اصلی را بیان می‌کند.

لم ۴-۱۰ مجموعه نقاط P در فضای \mathbb{R}^d داده شده است. یک جواب بهینه برای مسئله‌ی ۱- مرکز با z داده‌ی پرت برای مجموعه نقاط P ، با فرض $\delta^* \leq \alpha r^*$ ، یک $(2 + \frac{\alpha}{4})$ - تقریب برای مسئله‌ی ۲- مرکز با z داده‌ی پرت برای مجموعه‌ی نقاط P ارائه می‌دهد.



شکل ۴-۹: اثبات لم ۴-۱۰

اثبات. فرض کنید r_1^* و r^* به ترتیب شعاع بهینه برای مسئله‌ی ۱- مرکز و ۲- مرکز با z داده‌ی پرت برای مجموعه نقاط P باشد. به وضوح $r_1^* \leq r^*$ است، زیرا هر جواب درست B^* برای مسئله‌ی ۱- مرکز با z داده‌ی پرت، یک جواب درست (B^*, B^*) برای مسئله‌ی ۲- مرکز با z داده‌ی پرت ارائه می‌دهد. $B_1^*(c_1^*, r_1^*)$ و $B_2^*(c_2^*, r^*)$ را دو توپ جواب مسئله‌ی ۲- مرکز با z داده‌ی پرت در نظر بگیرید. c را نقطه‌ی وسط پاره‌خط واصل مراکز c_1^* و c_2^* در نظر بگیرید (همان‌طور که در شکل ۴-۹ نشان داده شده است). به وضوح، $B(c, \frac{\delta}{4} + 2r^*)$ هر دوی B_1^* و B_2^* را می‌پوشاند. بنابراین یک جواب قابل قبول برای مسئله‌ی ۱- مرکز با z داده‌ی پرت ارائه می‌دهند، و در نتیجه داریم:

$$r_1^* \leq (2 + \frac{\alpha}{4})r^*$$

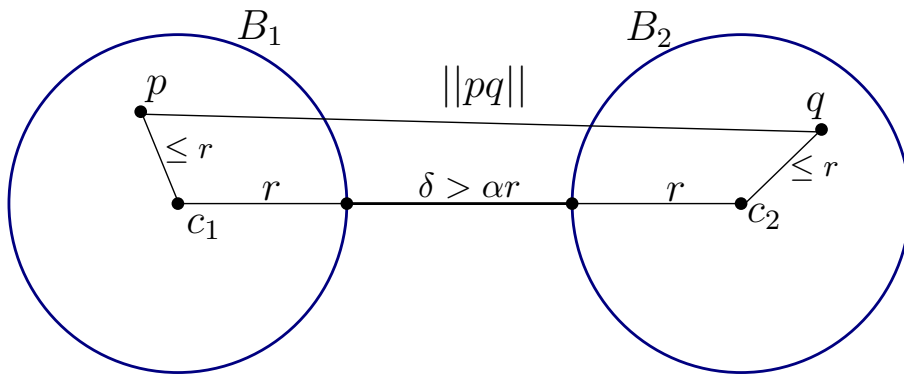
□

حال کافی است، به طور کاملاً مشابه با الگوریتم ۵ از الگوریتم ۴ برای تقریب r^* استفاده کنیم و با تقسیم بندی بازه‌ی $[0, 1/2r]$ به $m = \left\lceil \frac{1/\epsilon}{\epsilon} \times (4 + \alpha) \right\rceil$ زیر بازه، تمام گزینه‌های ممکن را امتحان کنیم. با استفاده از روش ارائه شده و با حذف فرض پرت نبودن داده‌ی اول جویبار داده به قضیه‌ی زیر می‌رسیم:

قضیه ۴-۱۱ اگر $\delta^* \leq \alpha r^*$ باشد، یک $(1/8 + \epsilon)$ -تقریب برای مسئله‌ی ۲-مرکز با z داده‌ی پرت با مصرف حافظه‌ی $O(\frac{dz^4}{\epsilon})$ و زمان به‌روزرسانی $O(\frac{dz^5}{\epsilon})$ قابل ارائه است.

۴-۳-۲ حالت $\delta^* > \alpha r^*$

در این بخش، الگوریتمی با ضریب تقریب $1/8 + \epsilon$ برای حالتی که دو توپ بهینه بیش از αr^* از یک‌دیگر فاصله دارند ارائه می‌دهیم. با دو مشاهده‌ی ساده شروع می‌کنیم:



شکل ۴-۱۰: اثبات مشاهده‌ی ۴-۱۲

مشاهده‌ی ۴-۱۲ فرض کنید که توپ B_1 و B_2 ، دو توپ با شعاع r باشند، به‌طوری‌که فاصله‌ای بیش‌تر از αr دارند. به ازای هر دو نقطه‌ی $p \in B_1$ و $q \in B_2$ ، داریم:

$$1 \leq \frac{\|pq\|}{\delta} < \frac{4 + \alpha}{\alpha}$$

اثبات. همان‌طور که در شکل ۴-۱۰ می‌بینید، با استفاده از نامساوی مثلثی رابطه‌ی زیر برقرار است:

$$\|pq\| \leq \|pc_1\| + \|c_1c_2\| + \|c_2q\| \leq r + r + \delta + r + r$$

از طرفی با توجه به نحوه‌ی تعریف δ ، می‌دانیم فاصله‌ی هر زوج دلخواه از (B_1, B_2) از جمله p و q ،

حداقل δ است. در نتیجه داریم:

$$\delta \leq \|pq\| \leq 4r + \delta < \frac{4\delta}{\alpha} + \delta$$

که با تقسیم طرفین بر δ به حکم مسئله می‌رسیم.

□

مشاهده‌ی ۴-۱۳ فرض کنید B_1 و B_2 دو توپ با فاصله‌ای برابر δ باشند و B یک توپ با شعاع کمتر از $\frac{\delta}{4}$ باشد. آنگاه B حداکثر با یکی از B_1 و B_2 تقاطع دارد.

در ادامه، تعدادی ویژگی برای توپ‌های بهینه‌ی B_1^* و B_2^* ارائه می‌دهیم.

لم ۴-۱۴ فرض کنید B_1^* و B_2^* دو توپ α -جداپذیر باشند ($\alpha > 4$). اگر p نقطه‌ای دلخواه از B_1^* و S زیرمجموعه‌ی $z+1$ عضو از P شامل دورترین نقاط از p باشد، آنگاه $S \cap B_1^*$ تهی است.

اثبات. از برهان خلف برای اثبات استفاده می‌کنیم. با برهان خلف فرض می‌کنیم که خلاف حکم مسئله برقرار و $S \cap B_1^*$ تهی باشد. چون $|S| = z+1 > z$ است، بنابراین عضوی از S وجود دارد که جزء داده‌های پرت در جواب بهینه نیست و چون S اشتراکش با B_1^* تهی است، بنابراین آن عضو داخل B_1^* قرار دارد و مجموعه‌ی $S \cap B_1^*$ تهی است. نقطه‌ی q را دورترین عضو $S \cap B_1^*$ در نظر بگیرید. توپ $B(p, \|pq\|)$ را در نظر بگیرید. به ازای هر نقطه‌ی $s \in P \setminus S$ ، $\|ps\| \leq \|pq\|$ است، زیرا $s \notin S$ و $q \in S$ است. بنابراین B ، مجموعه‌ی $P \setminus S$ را به طور کامل می‌پوشاند. از طرفی چون $p, q \in B_1^*$ است داریم:

$$\|pq\| \leq \|pc_1^*\| + \|c_1^*q\| \leq 2r^*$$

بنابراین با توجه به مشاهده‌ی ۴-۱۳، $B_1^* \cap B$ تهی است. بنابراین $B_1^* \cap P$ تهی است و در نتیجه B_1^* تهی است، که با بهینه بودن B_1^* و B_2^* تناقض دارد.

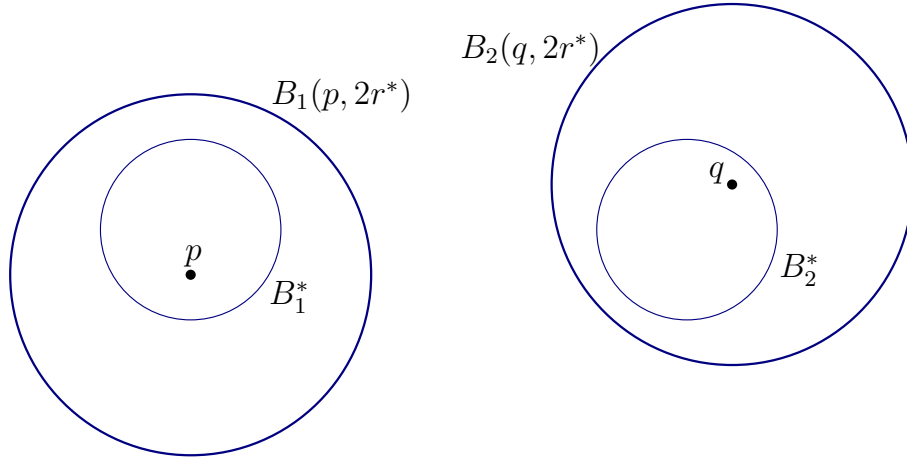
□

لم ۴-۱۵ فرض کنید p نقطه‌ای دلخواه در B_1^* باشد و $q, z+1$ - دورترین نقطه از p باشد. آنگاه $\delta^* > \frac{\alpha}{\alpha+4} \|pq\|$.

اثبات. با استفاده از لم ۴-۱۴، نقطه‌ی $q' \in B_p^*$ وجود دارد به طوری که $\|pq'\| \leq \|pq\|$. بنابراین با توجه به مشاهده‌ی ۴-۱۲ داریم:

$$\frac{\|pq\|}{\delta^*} \leq \frac{\|pq'\|}{\delta^*} < \frac{\alpha + 4}{\alpha}$$

□



شکل ۴-۱۱: اثبات لم ۴-۱۶

لم ۴-۱۶ دو نقطه‌ی $p \in B_1^*(c_1, r^*)$ و $q \in B_p^*(c_2, r^*)$ را در نظر بگیرید، آنگاه $B_1^* \subset B(p, 2r^*)$ و $B_p^* \subset B(q, 2r^*)$ است و در نتیجه حداکثر z نقطه از P خارج $B(p, 2r^*) \cup B(q, 2r^*)$ قرار می‌گیرد.

اثبات. نقطه‌ی دلخواه $p' \in B_1^*$ در نظر بگیرید. داریم:

$$\|pp'\| \leq \|pc_1\| + \|p'c_1\| \leq 2r^*$$

و در نتیجه $B_1^* \subset B(p, 2r^*)$ است (به شکل ۴-۱۱ مراجعه کنید). به طور مشابه ثابت می‌شود $B_p^* \subset B(q, 2r^*)$ است. با توجه به این که حداکثر z نقطه‌ی پرت خارج B_1^* و B_p^* قرار می‌گیرد، در نتیجه اثبات کامل است.

□

لم ۴-۱۷ فرض کنید S زیرمجموعه‌ای از P با اندازه‌ی حداقل $(z+1)(d+1)$ باشد که توسط تویی با شعاع کم‌تر از $\frac{\delta^*}{4}$ پوشانده می‌شود. آنگاه c_p نقطه‌ی مرکزی نقاط S ، یا داخل B_1^* قرار می‌گیرد یا داخل B_p^* قرار می‌گیرد.

اثبات. با توجه به این که اندازه‌ی S بیش‌تر از z است، حداقل یک نقطه‌ی غیر پرت داخل S قرار دارد. بنابراین با توجه به مشاهده‌ی ۴-۱۳، B دقیقاً با یکی از B_1^* یا B_2^* تقاطع دارد. بدون کم شدن از کلیت مسئله، فرض کنید B با توپ B_1^* تقاطع دارد. حال اگر $c_p \notin B_1^*$ نباشد، بنابراین طبق مشاهده‌ی ۴-۱ حداقل $1 + z$ نقطه‌ی دیگر خارج B_1^* قرار می‌گیرند، که با وجود حداکثر z داده‌ی پرت را نقض می‌کند. بنابراین فرض $c_p \notin B_1$ اشتباه بوده و حکم ثابت شد.

□

الگوریتم اصلی

در این بخش، الگوریتم اصلی برای حالتی که $\delta^* > \alpha r^*$ است را ارائه می‌دهیم. در هر لحظه، الگوریتم نقاط P از جویبار داده را به سه دسته‌ی مجزای B_1 ، B_2 و حافظه‌ی میان‌گیر Buffer افراز می‌کند. بدون کم شدن از کلیت مسئله، فرض کنید $p_1 \in B_1^*$ است. الگوریتم سعی می‌کند نقاط را به گونه‌ای به سه دسته افراز کند که B_1 به طور کامل B_1^* را بپوشاند و B_2 ، B_1^* را به طور کامل بپوشاند و Buffer تنها شامل تعدادی از نقاط پرت در جواب بهینه است. توجه کنید که B_1 و B_2 علاوه بر پوشش توپ متناظر در جواب بهینه، ممکن است تعدادی از نقاط پرت را نیز شامل شوند.

با شروع الگوریتم، مرکز توپ B_1 را که با c_1 نشان می‌دهیم برابر p_1 قرار داده می‌شود و c_2 را از میان نقاطی که تا کنون پردازش شده است به عنوان کاندیدا برای مرکز B_2 انتخاب می‌کند. الگوریتم همچنین دو متغیر δ و r را در طول جویبار داده به‌روزرسانی می‌کند به طوری که در هر لحظه، δ کران پایینی برای δ^* است و r تحت شرایطی که در ادامه گفته می‌شود، کران بالایی برای $2r^*$ است.

شبه کد الگوریتم، در الگوریتم ۷ آورده شده است. به محض ورود نقطه‌ی p از جویبار داده‌ی P ، الگوریتم ارائه شده، سعی می‌کند آن را به توپ B_1 یا B_2 اضافه کند. عمل اضافه کردن نقاط به توپ‌های B_1 و B_2 به ترتیب به وسیله‌ی توابع $\text{AddTo}B_1$ و $\text{AddTo}B_2$ انجام می‌شود. اگر نقطه‌ی p ، در هیچ کدام از توپ‌ها قرار نگیرد، به Buffer اضافه می‌شود. تابع $\text{AddTo}B_1$ نقطه‌ی p را به B_1 اضافه می‌کند اگر فاصله‌ی p از c_1 کم‌تر مساوی δ باشد. تابع $\text{AddTo}B_2$ نقطه‌ی p را به مجموعه‌ی B_2 اضافه می‌کند اگر نقطه‌ی p از c_2 حداکثر r فاصله داشته باشند. هر دوی توابع، مقدارهای δ و r را در صورت لزوم تغییر می‌دهند که ناوردهای داخل لم ۴-۱۸ برقرار بماند.

الگوریتم ۷ الگوریتم برای ۲-مرکز در حالت دور

- ۱: c_1 را برابر p_1 قرار بده.
 - ۲: r و δ را برابر صفر قرار بده.
 - ۳: به ازای هر نقطه‌ی $p \in P$:
 - ۴: اگر p_1 قابل اضافه شدن به B_1 و B_2 نبود: \Leftarrow از دو تابع $AddToB_1$ و $AddToB_2$ به ترتیب برای اضافه کردن نقطه به B_1 و B_2 استفاده می‌شود.
 - ۵: p را به Buffer اضافه کن.
 - ۶: تا وقتی $|Buffer| \geq z$:
 - ۷: اگر $|B_2| \geq (d+1)(z+1)$:
 - ۸: B_1 را برابر $B_1 \cup B_2$ قرار بده.
 - ۹: B_2 را برابر مجموعه‌ی تهی قرار بده.
 - ۱۰: در غیر این صورت:
 - ۱۱: اگر c_2 مشخص شده است:
 - ۱۲: c_2 را به B_1 اضافه کن.
 - ۱۳: متغیر محلی T را برابر $Buffer \cup B_2 \setminus \{c_2\}$ قرار بده.
 - ۱۴: B_2 را تهی قرار بده.
 - ۱۵: c_2 را $(z+1)$ -امین دورترین نقطه از c_1 در T قرار بده.
 - ۱۶: r را برابر با $\frac{2}{\alpha} \|c_1 c_2\|$ قرار بده.
 - ۱۷: به ازای هر $p \in T$:
 - ۱۸: p را به B_2 اضافه کن.
 - ۱۹: Buffer را برابر $B_2 \setminus T$ قرار بده.
-

الگوریتم ۸ تابع اضافه‌کننده‌ی نقطه به B_1

- ۱: اگر حداقل $1 + z$ نقطه پردازش شده است:
- ۲: q را $(z + 1) -$ دورترین نقطه از c_1 در نقاطی که تا کنون آمده‌اند در نظر بگیر.
- ۳: در غیر این صورت:
- ۴: q را برابر c_1 در نظر بگیرد.
- ۵: δ را برابر $\frac{\alpha}{\alpha+4} \|c_1 q\|$ قرار بده.
- ۶: اگر $p \in B(c_1, \delta)$:
- ۷: p را به مجموعه‌ی B_1 اضافه کن.
- ۸: برگردان true
- ۹: برگردان false

زمانی که Buffer سرریز می‌شود (در الگوریتم ۷)، الگوریتم یکی از دو عملیات زیر را وابسته به این‌که اندازه‌ی B_2 چقدر است، انجام می‌دهد. اگر اندازه‌ی $|B_2|$ بزرگ‌تر مساوی $(z + 1)(d + 1)$ باشد، تمام اعضای B_2 به B_1 اضافه می‌شود و B_2 خالی می‌گردد. در غیر این صورت، c_2 در صورتی که قبلاً تعیین شده باشد، به B_1 اضافه می‌شود و نقطه‌ی دیگری از $B_2 \cup \text{Buffer} \setminus \{c_2\}$ به عنوان c_2 جدید انتخاب می‌گردد. حلقه‌ی مذکور، حداکثر $O(dz)$ بار اجرا می‌شود، زیرا پس از اولین اجرا، مطمئن هستیم که T حداکثر $z + (z + 1)(d + 1)$ نقطه دارد و در هر مرحله حداقل یک عنصر از آن حذف می‌گردد (c_2). توجه داشته باشید که هر نقطه حداکثر یک‌بار به عنوان c_2 انتخاب می‌شود، بنابراین حلقه‌ی مذکور، به طور سرشکن^{۱۲} یک‌بار به ازای هر نقطه از جویبار داده اجرا می‌شود.

برای تحلیل، علاوه بر c_2 ، نیاز به نقطه‌ی مرکزی به نام c_p داریم. زمانی که $|B_2| < (z + 1)(d + 1)$ باشد، آنگاه c_p همان c_2 است و در غیر این صورت، c_p را نقطه‌ی مرکزی $(z + 1)(d + 1)$ نقطه‌ی اولی که به B_2 اضافه می‌شوند قرار داده می‌شود.

لم ۴-۱۸ ثابت‌های حلقه‌ی زیر در طول اجرای الگوریتم حفظ می‌شوند:

^{۱۲} Amortized

الگوریتم ۹ تابع اضافه‌کننده نقطه به B_2

- ۱: اگر c_2 تعیین شده باشد و $p \in B(c_2, r)$:
- ۲: p را به B_2 اضافه کن.
- ۳: اگر $|B_2| = (d+1)(z+1)$:
- ۴: r را $(2 + \frac{2}{\alpha})$ برابر کن.
- ۵: به ازای $p \in B(c_2, r)$:
- ۶: اگر $p \in B(c_2, r)$:
- ۷: p را به B_2 اضافه کن.
- ۸: p را از Buffer حذف کن.
- ۹: در غیر این صورت:
- ۱۰: برگردان true
- ۱۱: برگردان false

$$1. \delta < \delta^*$$

$$2. r \leq \frac{\delta}{4}$$

$$3. B_1 \cap B_2^* = \emptyset$$

$$4. \text{ اگر } c_p \in B_2^* \text{ باشد، آن‌گاه:}$$

$$2r^* \leq r \quad (\bar{I})$$

$$(ب) \quad B_2 \cap B_2^* = \emptyset$$

(ج) تمام نقاط داخل Buffer در جواب بهینه داده‌ی پرت هستند.

اثبات. ۱. در ابتدای اجرای الگوریتم $\delta = 0$ است که به وضوح حکم برقرار است. بعد از این که $z+1$ نقطه از جویبار داده پردازش می‌شود، تابع $AddToB_1$ مقدار δ را به $\frac{\alpha}{\alpha+4} \|c_1 q\|$ افزایش می‌دهد، که در آن $q, (z+1)$ - دورترین نقطه از c_2 در جویبار داده است. از طرفی چون $c_1 \in B_1^*$ است، طبق **لم ۴-۱۵** داریم $\delta < \delta^*$.

۲. زمانی که متغیر c_2 در الگوریتم ۷ تعیین می‌شود، $(z+1)$ - دورترین نقطه از c_1 در بین اعضای $T \subset P$ است و r برابر $\frac{2}{\alpha} \|c_1 c_2\|$ است. اگر q ، $(z+1)$ - دورترین نقطه از c_1 در جویبار داده در آن لحظه باشد، آنگاه $\|c_1 c_2\| \leq \|c_1 q\|$ است. با فرض این‌که $\alpha \geq 16$ باشد، داریم:

$$\frac{2}{\alpha} \|c_1 c_2\| \leq \frac{1}{6} \times \frac{\alpha \|c_1 q\|}{\alpha + 4} \leq \frac{\delta}{6}$$

و در نتیجه:

$$r \leq \left(2 + \frac{2}{\alpha}\right) \times \frac{2}{\alpha} \|c_1 c_2\| \leq 3 \times \frac{2}{\alpha} \|c_1 c_2\| \leq \frac{\delta}{2}$$

که نشان می‌دهد که صورت ناوردا درست است، حتی بعد از افزایشی که در تابع $AddToB_2$ می‌یابد.

۳. در ابتدا لم زیر را ثابت می‌کنیم:

ادعای ۴-۱۹ اگر c_2 تعیین شده باشد، آنگاه $B_2(c_2, r) \subset B(c_p, \frac{2}{\alpha} \|c_1 c_p\|)$ است.

اثبات. اگر $|B_2| < (d+1)(z+1)$ باشد، با توجه به این‌که $c_p = c_2$ است و $r = \frac{2}{\alpha} \|c_1 c_2\|$ ، در نتیجه $B_2 = B(c_p, \frac{2}{\alpha} \|c_1 c_p\|)$ است و حکم برقرار است. در حالتی که اندازه‌ی B_2 به $(d+1)(z+1)$ می‌رسد، نقطه‌ی c_p به نقطه‌ی مرکزی نقاط B_2 انتقال می‌یابد و r ، $(2 + \frac{2}{\alpha})$ برابر می‌گردد. از طرفی چون نقطه‌ی مرکزی نقاط درون B_2 ، داخل B_2 قرار می‌گیرد، بنابراین $c_p \in B(c_2, \frac{2}{\alpha} \|c_1, c_2\|)$ است. در نتیجه داریم:

$$\|c_2 c_p\| \leq \frac{2}{\alpha} \|c_1 c_2\|$$

و در نتیجه:

$$\frac{2}{\alpha} \|c_1 c_p\| \leq \frac{2(\|c_1 c_2\| + \|c_2 c_p\|)}{\alpha} \leq \frac{2}{\alpha} \|c_1 c_2\| \left(1 + \frac{2}{\alpha}\right)$$

که نتیجه می‌دهد:

$$B(c_p, \frac{2}{\alpha} \|c_1 c_p\|) \subset B_2(c_2, \frac{2}{\alpha} \|c_1 c_2\| (2 + \frac{2}{\alpha}))$$

□

حال با استفاده از ادعای ۴-۱۹، حکم را ثابت می‌کنیم. نقطه‌ی p از جویبار داده را در نظر بگیرید که به B_1 اضافه شده است. این نقطه در دو شرایط می‌تواند به B_1 اضافه شده باشد. حالت اول،

زمانی است که تابع $AddToB_1$ صدا زده می‌شود. در این تابع، نقطه‌ی p تنها زمانی به B_1 اضافه می‌شود که در فاصله‌ی δ از c_1 قرار داشته باشد. با توجه به ناوردای ۱، مطمئن هستیم که $\delta < \delta^*$ و در نتیجه $\|c_1 p\| < \delta^*$ است و در نتیجه $p \notin B_1^*$. در حالت دوم، در الگوریتم ۷، زمانی که حافظه‌ی میان‌گیر $Buffer$ سرریز می‌کند و B_2 خالی نیست، الگوریتم وابسته به اندازه‌ی B_2 عمل می‌کند. اگر اندازه‌ی B_2 هنوز به $(z+1)(d+1)$ نرسیده باشد، آنگاه $c_p = c_2$ است. در این حالت، الگوریتم ۷، c_2 را به B_1 اضافه می‌کند. با فرض خلف فرض کنید که $c_p \in B_1^*$ باشد. آنگاه با توجه ناوردای ۲ و ۴ قسمت (آ) داریم:

$$2r^* \leq r \leq \frac{\delta}{2} < \frac{\delta^*}{2}$$

در نتیجه با توجه به لم ۴-۱۶، حداکثر باید z نقطه خارج $B_1 \cup B_2$ قرار بگیرد، که با سرریز شدن حافظه‌ی میان‌گیر $Buffer$ تناقض دارد. در حالتی که $|B_2| \geq (d+1)(z+1)$ است، آنگاه c_p ، نقطه‌ی مرکزی $(d+1)(z+1)$ اولین نقاطی است که به B_2 اضافه شده‌اند. در این حالت، تمام نقاط B_2 به B_1 اضافه می‌شود. با استفاده از ناوردای ۲، داریم:

$$r \leq \frac{\delta}{2} < \frac{\delta^*}{2}$$

در نتیجه، طبق لم ۴-۱۷، $c_p \in B_1^*$ یا $c_p \in B_2^*$ است. با فرض خلف، فرض کنید که $c_p \in B_2^*$ است. در این حالت، با توجه به ناوردای ۴ قسمت (آ)، و ادعای ۴-۱۹، توپ $B(c_p, \frac{2}{\alpha} \|c_1 c_p\|)$ را می‌پوشاند و $2r^* \leq r$ است. بنابراین کاملاً مشابه حالتی که $|B_2| < (d+1)(z+1)$ ، با سرریز شدن حافظه‌ی میان‌گیر $Buffer$ تناقض دارد و در نتیجه $c_p \in B_1^*$ و در نتیجه $B_2 \cap B_1^* = \emptyset$ است و اضافه کردن آن به B_1 صورت ناوردا را نقض نمی‌کند.

۴. (آ) اگر $c_1 \in B_1^*$ باشد و $c_p \in B_2^*$ باشد، طبق مشاهده‌ی ۴-۱۳، آنگاه

$$1 \leq \frac{\|c_1 c_p\|}{\delta^*} \leq \frac{\|c_1 c_p\|}{\alpha r^*}$$

و در نتیجه:

$$2r^* \leq \frac{2}{\alpha} \|c_1 c_p\|$$

اگر $|B_2| < (d+1)(z+1)$ باشد، $c_p = c_2$ است و با توجه به الگوریتم ۷، $r = \frac{2}{\alpha} \|c_1 c_2\|$

است و در نتیجه $2r^* \leq r$ است. اگر $|B_2| \geq (d+1)(z+1)$ باشد، مشابه با ناوردای ۲

داریم:

$$2r^* \leq \frac{2}{\alpha} \|c_1 c_p\| \leq (1 + \frac{2}{\alpha}) \frac{2}{\alpha} \|c_1 c_2\| \leq (2 + \frac{2}{\alpha}) \frac{2}{\alpha} \|c_1 c_2\| = r$$

(ب) بر اساس ناوردای ۴ قسمت (آ)، اگر $c_p \in B_1^*$ باشد، در نتیجه داریم:

$$2r^* \leq r \leq \frac{\delta}{2}$$

و $c_p \in B_2$ است. حال با توجه به ناوردای ۱ و مشاهده‌ی ۴-۱۳، B_2 تنها B_1^* را قطع

می‌کند و در نتیجه $B_2 \cap B_1^* = \emptyset$.

(ج) با توجه به ناوردای ۳ و ناوردای ۴ قسمت (آ) داریم:

$$2r^* \leq r \leq \frac{\delta}{2} < \frac{\delta^*}{2}$$

و در نتیجه با توجه به لم ۴-۱۶، تمام نقاط داخل حافظه‌ی میان‌گیر Buffer یا خارج از

$B_1 \cup B_2$ داده‌ی پرت هستند.

□

پاسخ‌گویی به پرسمان‌ها

در این قسمت، نشان می‌دهیم با تقسیم‌بندی که الگوریتم ۷ در طول اجرای الگوریتم نگه می‌دارد، چگونه به پرسمان‌هایی همانند پرسمان زیر پاسخ می‌دهد.

- اگر بدانیم دو توپ بهینه‌ی جواب مسئله‌ی ۲- مرکز با z داده‌ی پرت، α - جداپذیر باشند، دو توپ هم شعاع پیدا کنید که همه‌ی نقاط به غیر از حداکثر z نقطه از نقاطی که تاکنون در جویبار داده آمده‌اند را بپوشانند.

الگوریتم پاسخ‌گویی به پرسمان در الگوریتم ۱۰ آمده است. ایده‌ی اصلی پاسخ‌گویی به پرسمان استفاده از تقسیم‌بندی که الگوریتم ۷ ارائه می‌دهد، است. با توجه به فرض‌های اولیه و با استفاده از ناوردهای ۳ و ۴، اگر $c_p \in B_1^*$ باشد، آنگاه B_1 به طور کامل B_1^* را می‌پوشاند و B_2 به طور کامل B_2^* را می‌پوشاند. اما ممکن است فرض $c_p \in B_1^*$ اشتباه باشد و در نتیجه B_1 ، B_2^* و B_2 ، B_1^* را نپوشانند. برای برطرف کردن این مشکل، تمام حالت‌های ممکن برای c_2 (که از روی آن تمام حالت‌های c_p به

دست می‌آید) امتحان می‌کنیم و به ازای آن تقسیم‌بندی B_1, B_2 و Buffer را به دست آورده و از روی آن با استفاده از الگوریتم ۱۱، یک جواب برای مسئله‌ی ۲-مرکز با حداکثر z داده‌ی پرت ارائه می‌دهد.

الگوریتم ۱۰ پاسخ‌گویی به پرسمان

۱: مجموعه‌ی solutions را برابر $\{\text{MinCover}(B_1, B_2, \text{Buffer})\}$ قرار بده.

۲: اگر $|B_2| \geq (d+1)(z+1)$ است:

۳: مجموعه‌ی candidates را برابر Buffer قرار بده.

۴: مجموعه‌ی B_1 را برابر $B_1 \cup B_2$ قرار بده.

۵: مجموعه‌ی B_2 را تهی کن.

۶: در غیر این صورت:

۷: مجموعه‌ی candidates را برابر $B_2 \cup \text{Buffer} \setminus \{c_2\}$ قرار بده.

۸: به ازای هر $c \in \text{candidates}$:

۹: r را برابر $\frac{\gamma}{\alpha} \|c_1 c\|$ قرار بده.

۱۰: B'_1 را برابر B_1 قرار بده.

۱۱: δ را برابر $\max\{\delta_0, r\}$ قرار بده.

۱۲: B'_2 و حافظه‌ی میان‌گیر Buffer' را برابر مجموعه‌ی تهی قرار بده.

۱۳: به ازای هر $p \in \text{candidates}$:

۱۴: اگر نقطه‌ی p به B'_1 و B'_2 اضافه نشد:

۱۵: p را به حافظه میان‌گیر Buffer' اضافه کن.

۱۶: $\text{MinCover}(B'_1, B'_2, \text{Buffer}')$ را به مجموعه‌ی solutions اضافه کن.

۱۷: کم‌ترین عضو مجموعه‌ی solutions را برگردان.

فرض کنید C مجموعه‌ی نقاط کاندیدا برای c_2 باشد. با استفاده از ناوردای ۳، داریم $B_1 \cap B_2^* = \emptyset$.

بنابراین، $(B_2 \cup \text{Buffer}) \cap B_2^* \neq \emptyset$

و در نتیجه وجود دارد c_2 در این مجموعه که می‌توان از روی آن، جواب با تقریب مناسبی برای ۲-

مرکز با حداکثر z داده‌ی پرت به دست آورد. در لم ۴-۲۰، برای حالتی $|B_2| \geq (d+1)(z+1)$ است،

الگوریتم ۱۱ محاسبه‌ی پوشش بهینه

- ورودی: B_1 به عنوان توپی که با B^* اشتراک ندارد، B_2 به عنوان توپی که با B^* اشتراک ندارد و Buffer
- به عنوان زیر مجموعه‌ای از نقاط پرت در جواب بهینه.
- ۱: solutions را مجموعه‌ی تهی قرار دهید.
- ۲: به ازای k در بازه‌ی $[0 \dots (z - |Buffer|)]$:
- ۳: r_1 را برابر شعاع $1 - \text{Center}(B_1, k)$ قرار بده.
- ۴: r_2 را برابر شعاع $1 - \text{Center}(B_2, z - |Buffer| - k)$ قرار بده.
- ۵: بیشینه‌ی r_1 و r_2 را به مجموعه‌ی solutions اضافه کن.
- ۶: کمینه عضو solutions را به عنوان خروجی برگردان.

نشان داده شده است که $B^* \cap (B_2 \cup \{c_2\}) \neq \emptyset$. بنابراین اگر $(z+1)(d+1) \geq |B_2|$ باشد، کافی است مجموعه‌ی $B_2 \cup \{c_2\}$ را به عنوان مجموعه‌ی نقاط کاندیدا برای c_2 در نظر گرفت. بنابراین اندازه‌ی مجموعه‌ی نقاط کاندیدا از $\mathcal{O}(zd)$ است.

لم ۴-۲۰ در هر لحظه، اگر $(z+1)(d+1) \geq |B_2|$ باشد و $c_p \notin B^*$ باشد، آنگاه $B_2 \cap B^* = \emptyset$.

اثبات. با استفاده از ناوردهای ۱ و ۲، داریم:

$$r \leq \frac{\delta}{2} < \frac{\delta^*}{2}$$

با توجه به لم ۴-۱۷، $c_p \in B_1^* \cup B_2^*$ است. اگر $c_p \notin B_2^*$ باشد، در نتیجه $c_p \in B_1^*$ است. و در نتیجه با توجه به مشاهده‌ی ۴-۱۳، B_2 حداکثر یکی از B_1^* و B_2^* را قطع می‌کند و در نتیجه $B_2 \cap B^* = \emptyset$.

□

الگوریتم پاسخ‌گویی به پرسمان، به ازای هر نقطه‌ی $c \in C$ ، دو توپ $(c_1, \max\{\delta, \frac{r}{\alpha}\|c_1 c\|\})$ و $B'_1(c, \frac{r}{\alpha}\|c_1 c\|)$ را تشکیل می‌دهد. اگر نقطه‌ی کاندیدا برابر c_2 کنونی باشد، $B'_1 = B_1$ است. از طرفی چون $\frac{r}{\alpha}\|c_1 c_2\| \leq r \leq \frac{\delta}{2}$ است، با توجه به ناوردهای ۲، $B'_1 \subset B_2$ است و در نتیجه نیازی به ساخت B'_1 و B'_2 نیست. اگر نقطه‌ی کاندیدا برابر c_2 نباشد، با توجه به ناوردهای ۲، مطمئن هستیم که $B_1 \subset B'_1$ است. بنابراین کافی است ببینیم کدام یک از نقاط $B_1 \cup \text{Buffer}$ داخل B'_1 قرار می‌گیرند.

اگر $|B_2| \geq (d+1)(z+1)$ باشد، بدان معناست که $c_p \notin B_2^*$ است. با توجه به لم ۴-۲۰، B_2 را می‌توان بدون نقض کردن ناوردای ۳، به B_1' اضافه کرد. بنابراین در این حالت، کافی است نقاط Buffer برای اضافه شدن به B_1' بررسی شوند. الگوریتم ۱۰ از دو تابع $\text{AddTo}B_1'$ و $\text{AddTo}B_2'$ به ترتیب برای اضافه کردن نقاط به B_1' و B_2' استفاده می‌کند. این دو تابع، کاملاً مشابه $\text{AddTo}B_1$ و $\text{AddTo}B_2$ است، با این تفاوت که به ترتیب نقاط را به B_1' و B_2' اضافه می‌کند. از آنجایی که الگوریتم ۱۰، تمام گزینه‌های ممکن برای کاندیدها را بررسی می‌کند، حداقل یک $c^* \in C$ وجود دارد که $c^* \in B_2^*$ برقرار است. B_2' متناظر با نقطه‌ی کاندیدای c^* را به ترتیب، B'' و B_2'' نشان می‌دهیم. با توجه به مشاهده‌ی ۴-۱۲ داریم:

$$1 \leq \frac{\|c_1 c^*\|}{\alpha^* r^*} < \frac{\|c_1 c^*\|}{\alpha r^*}$$

و از طرفی داریم:

$$2r^* \leq \frac{2}{\alpha} \|c_1 c^*\|$$

و در نتیجه با توجه به لم ۴-۱۶، $B_1^* \subset B_1''$ و $B_2^* \subset B_2''$ است. که در واقع بدان معناست که B'' به طور کامل B_1^* را می‌پوشاند و B_2'' به طور کامل B_2^* را می‌پوشاند و تمام نقاط داخل Buffer در جواب بهینه داده‌ی پرت هستند. تنها نقطه‌ی ابهام باقی‌مانده، عدم اطلاع از تقسیم‌بندی نقاط پرت بین B_1'' و B_2'' است که الگوریتم ۱۱ تمام حالت‌های ممکن برای این تقسیم‌بندی را امتحان کرده و کم‌ترین شعاع در بین تمام حالات را به عنوان جواب نهایی ارائه می‌دهد.

قضیه ۴-۲۱ الگوریتم ۱۰، در حالتی که دو توپ بهینه α - جداپذیر باشند، می‌تواند جوابی با ضریب تقریب $\epsilon + 1/8$ برای مسئله‌ی ۲- مرکز با z داده‌ی پرت در زمان اجرای $O(\frac{z^4 d^3}{\epsilon})$ ارائه دهد.

اثبات. همان‌طور که گفته شد، تعداد کاندیدهای موجود از مرتبه‌ی $O(zd)$ است. بنابراین الگوریتم ۱۰، $O(zd)$ بار B_1' و B_2' را حساب کرده و با آن، الگوریتم ۱۱ را صدا می‌زند. اگر از الگوریتم ۵ برای نگه‌داری B_1' و B_2' استفاده کنیم، محاسبه‌ی B_1' و B_2' از مرتبه‌ی $O(\frac{z^2 d}{\epsilon}) \times O(zd)$ زمان می‌برد (توجه کنید که فرض کرده‌ایم نقطه‌ی اول B_1' و B_2' که مرکز دسته‌ها هستند داده‌ی پرت نیستند) و اجرای الگوریتم ۱۱، از مرتبه‌ی $O(z) \times O(\frac{z}{\epsilon})$ زمان می‌برد. بنابراین زمان پاسخ‌گویی به پرسمان از مرتبه‌ی $O(\frac{z^4 d^3}{\epsilon})$ زمان خواهد برد.

□

و در نهایت قضیه‌ی زیر نتیجه می‌شود:

قضیه‌ی ۴-۲۲ در حالتی که $\delta^* > \alpha r^*$ است، یک $\epsilon + 1/8$ - تقریب برای مسئله‌ی ۲ - مرکز با z داده‌ی پرت قابل نگهداری است که از مرتبه‌ی $O(\frac{dz^4}{\epsilon})$ حافظه مصرف کرده و زمان به‌روزرسانی آن، از مرتبه‌ی $O(\frac{dz^5}{\epsilon})$ بوده و با مصرف $O(\frac{dz^5}{\epsilon})$ به پرسمان‌ها پاسخ می‌دهد.

اثبات. همان‌طور که در قسمت قبلی نشان داده شد، برای پاسخ‌گویی به پرسمان، الگوریتم ۱۰ از تقسیم‌بندی B_1 ، B_2 و $Buffer$ برای محاسبه‌ی جواب استفاده می‌کند. در حالت جویبار داده، امکان نگهداری تمام نقاط B_1 و B_2 وجود ندارد. بنابراین از داده‌ساختاری برای نگهداری مجموعه هسته‌ای از نقاط داخل B_1 و B_2 استفاده می‌کنیم، که قابلیت اضافه شدن نقطه و ارائه‌ی یک β - تقریب برای مسئله‌ی ۱ - مرکز با k داده‌ی پرت (برای k در بازه‌ی $[0, z]$) داشته باشد. از طرفی به علت نیاز به اضافه کردن کل B_2 به B_1 در بعضی از مراحل الگوریتم، $B_u = B_1 \cup B_2$ را موازی با B_1 و B_2 ، نگهداری می‌شود. توجه کنید که داده‌ساختارهای مورد نیاز برای B_1 و B_2 نیازی به نگهداری کل نقاط ندارند، بلکه تنها نیاز به یک حافظه‌ی میان‌گیر برای نگهداری $(d+1)(z+1)$ آخرین نقاطی که به داده‌ساختار اضافه شده‌اند دارد.

برای نگهداری B_1 و B_2 و B_u از الگوریتم جویبار داده‌ی ۵ ارائه شده در همین پایان‌نامه استفاده می‌کنیم، که یک الگوریتم با ضریب تقریب $\epsilon + 1/8$ ارائه می‌دهد. با توجه به الگوریتم ۵، حافظه‌ی مصرفی برابر $O(z) \times O(\frac{dz^2}{\epsilon})$ است (فرض کرده‌ایم نقاط اول هر دسته داده‌ی پرت نیست). زمان به‌روزرسانی نیز با توجه به اجرا شدن یک بار حلقه به صورت سرشکن در هر مرحله از مرتبه‌ی $O(\frac{dz^2}{\epsilon}) \times O(z) \times O(dz)$ زمان می‌برد. از طرفی با توجه به این‌که برای حذف فرض پرت نبودن نقطه‌ی اول نیاز به اجرای z نمونه موازی از الگوریتم داریم، بنابراین تمام تحلیل‌ها z برابر می‌شوند.

□

اگر در هر لحظه یک نمونه از الگوریتمی که برای حالت $\delta \leq \alpha r^*$ و حالت $\delta^* > \alpha r^*$ ارائه می‌دهیم را به طور موازی اجرا کنیم و به ازای هر پرسمان، برای هر دو حالت، جواب را به دست آورده و جواب با شعاع کم‌تر را به عنوان جواب نهایی بدسیم، آنگاه با توجه به درستی یکی از حالات در هر لحظه، جواب نهایی یک $\epsilon + 1/8$ برای جواب نهایی است.

قضیه‌ی ۴-۲۳ الگوریتم ارائه شده در این بخش، یک الگوریتم با ضریب تقریب $\epsilon + 1/8$ است که

با مصرف حافظه $O(\frac{dz^4}{\epsilon})$ و زمان به روزرسانی $O(\frac{dz^5}{\epsilon})$ ، مجموعه‌ای هسته برای مسئله‌ی ۲-مرکز با متریک اقلیدسی است.

فصل ۵

نتیجه‌گیری

در این پایان‌نامه گونه‌های مختلفی از دو مسئله‌ی ۱- مرکز و ۲- مرکز در حالت جویبار داده مورد بررسی قرار گرفت. همان‌طور که بیان شد این دو مسئله و حالت کلی آن، استفاده‌ی زیادی در علوم کامپیوتر دارند. از طرفی به علت افزایش روز افزون داده‌ها مدل جویبار داده‌ی مسئله در عمل بسیار کاربردی است.

در این پایان‌نامه ابتدا مسئله‌ی ۱- مرکز با داده‌ی پرت در حالت جویبار داده ارائه گردید. برای این مسئله دو الگوریتم متفاوت ارائه گردید. الگوریتم اول، با مصرف حافظه‌ی $O(z^2 d)$ جوابی با ضریب تقریب ۲ برای مسئله‌ی ۱- مرکز با داده‌ی پرت ارائه می‌دهد. الگوریتم ارائه شده، نسبت به الگوریتم ضربایی زاده [۱۳] الگوریتمی ساده‌تر است و در صورتی که d از z بزرگ‌تر باشد، حافظه‌ی مصرفی را نیز کاهش می‌دهد. در الگوریتم دوم، با استفاده از ایده‌ی مطرح شده در مرجع [۹]، الگوریتمی با ضریب تقریب $\epsilon + 1/8$ و حافظه‌ی مصرفی $O(\frac{dz^2}{\epsilon})$ ارائه شده است.

در بخش دوم، مسئله‌ی ۱- مرکز پوشاننده بدون داده‌ی پرت مورد بررسی قرار گرفته است. برای این مسئله، یک الگوریتم با ضریب تقریب $\epsilon + 1/8$ و حافظه‌ی مصرفی $O(\frac{d}{\epsilon})$ ارائه شد. در ادامه، با استفاده از الگوریتم [۱۰]، یک الگوریتم با ضریب تقریب $1/7$ و حافظه‌ی مصرفی $O(d)$ ارائه گردید. با استفاده از همین الگوریتم، در بخش بعدی برای مسئله‌ی ۱- مرکز با z داده‌ی پرت، الگوریتمی با ضریب تقریب $1/7$ با حافظه‌ی مصرفی $O(d \times (d+1)^z)$ ارائه شده است.

در نهایت، در بخش سوم، مسئله‌ی ۲- مرکز با z داده‌ی پرت مورد بررسی قرار گرفته و الگوریتمی با

ضریب تقریب $\epsilon + 1/8$ و حافظه‌ی مصرفی $O(\frac{dz^4}{\epsilon})$ ارائه شد که نسبت به الگوریتم پیشین برای k کلی، با ضریب تقریب $\epsilon + 4$ بهبود قابل توجهی محسوب می‌شود.

۵-۱ کارهای آتی

همان‌طور که در بخش قبلی به آن اشاره شد، دو الگوریتم با ضریب تقریب $\epsilon + 1/8$ برای مسئله‌ی ۱-مرکز و ۲-مرکز با z داده‌ی پرت ارائه شد. حدسی که وجود دارد، امکان تعمیم دو الگوریتم داده شده به الگوریتمی کلی برای مسئله‌ی k -مرکز با z داده‌ی پرت به ازای k دلخواه است.

از طرفی دیگر، هیچ کران پایینی دقیق‌تر از کران پایین $\frac{1+\sqrt{2}}{4}$ که برای مسئله‌ی ۱-مرکز به وسیله‌ی آگاروال و شاراف‌کومار ارائه شده است [۱۰] برای مسئله‌ی ۱-مرکز و ۲-مرکز با z داده‌ی پرت وجود ندارد. بنابراین حوزه‌ای که امکان بهبود دارد، کم کردن فاصله‌ی بین $1/8$ (بهترین الگوریتم موجود) و $\frac{1+\sqrt{2}}{4}$ (بهترین کران پایین) است که ممکن است با اثبات کران پایین بالاتر یا ارائه‌ی الگوریتم جدیدی که ضریب تقریب کم‌تر از $1/8$ داشته باشد ممکن گردد.

از طرفی ممکن است، بتوان الگوریتم موجود را بدون تغییر ضریب تقریب، از لحاظ میزان حافظه‌ی مصرفی، میزان زمان مورد نیاز برای به‌روزرسانی و زمان مورد نیاز برای پاسخ‌گویی به پرسشان بهبود بخشید.

مسئله‌ی k -مرکز پوشاننده به عنوان مسئله‌ای که کم‌تر مورد توجه قرار گرفته است را نیز مورد بررسی بیشتری قرار داد و الگوریتم‌های بهتری از لحاظ ضریب تقریب یا حافظه‌ی مصرفی و زمان به‌روزرسانی ارائه داد. از طرفی در حال حاضر، کران پایینی غیر از ضریب تقریب $\frac{1+\sqrt{2}}{4}$ برای این مسئله وجود ندارد که ممکن است بتوان آن را بهبود بخشید.

کتاب نامه

- [1] J. Han and M. Kamber. *Data Mining, Southeast Asia Edition: Concepts and Techniques*. Morgan kaufmann, 2006.
- [2] C. C. Aggarwal. *Data streams: models and algorithms*, volume 31. Springer Science & Business Media, 2007.
- [3] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms*, pages 642–651, 2001.
- [4] M. R. Garey and D. S. Johnson. Computers and intractability: a guide to the theory of NP-completeness. *WH Freeman & Co., San Francisco*, 1979.
- [5] N. Megiddo and K. J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal on Computing*, 13(1):182–196, 1984.
- [6] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., 2001.
- [7] R. M. McCutchen and S. Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In *International Workshop on Approximation Algorithms*, pages 165–178. 2008.
- [8] S. Guha. Tight results for clustering and summarizing data streams. In *Proceedings of the 12th International Conference on Database Theory*, pages 268–275, 2009.
- [9] H.-K. Ahn, H.-S. Kim, S.-S. Kim, and W. Son. Computing k centers over streaming data for small k. *International Journal of Computational Geometry and Applications*, 24(02):107–123, 2014.

- [10] P. K. Agarwal and R. Sharathkumar. Streaming algorithms for extent problems in high dimensions. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms*, pages 1481–1489, 2010.
- [11] T. M. Chan and V. Pathak. Streaming and dynamic algorithms for minimum enclosing balls in high dimensions. *Computational Geometry: Theory and Applications*, 47(2):240–247, 2014.
- [12] S.-S. Kim and H.-K. Ahn. An improved data stream algorithm for clustering. In *Proceedings of the 11th Latin American Theoretical Informatics Symposium*, pages 273–284. 2014.
- [13] 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.
- [14] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.
- [15] V. Estivill-Castro. Why so many clustering algorithms: a position paper. *ACM SIGKDD explorations newsletter*, 4(1):65–75, 2002.
- [16] M. Bern and D. Eppstein. Approximation algorithms for NP-hard problems. chapter Approximation Algorithms for Geometric Problems, pages 296–345. PWS Publishing Co., 1997.
- [17] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *Journal of the ACM*, 51(4):606–635, 2004.
- [18] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.
- [19] P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.
- [20] N. Megiddo. On the complexity of some geometric problems in unbounded dimension. *Journal of Symbolic Computation*, 10(3):327–334, 1990.
- [21] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Journal of Algorithms*, 21(3):579–597, 1996.

- [22] T. M. Chan. More planar two-center algorithms. *Computational Geometry: Theory and Applications*, 13(3):189–198, 1999.
- [23] P. K. Agarwal, R. B. Avraham, and M. Sharir. The 2-center problem in three dimensions. *Computational Geometry*, 46(6):734–746, 2013.
- [24] H. Zarrabi-Zadeh. Core-preserving algorithms. In *Proceedings of the 20th Canadian Conference on Computational Geometry*, pages 159–162, 2008.
- [25] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 626–635. ACM, 1997.
- [26] H. Zarrabi-Zadeh and T. M. Chan. A simple streaming algorithm for minimum enclosing balls. In *Proceedings of the 18th Canadian Conference on Computational Geometry*, pages 139–142, 2006.
- [27] M. Badoiu and K. L. Clarkson. Smaller core-sets for balls. In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms*, pages 801–802, 2003.
- [28] L. Danzer, B. Gruenbaum, and V. Klee. Helly’s theorem and its relatives. In *Proceedings of the Symposia in Pure Mathematics*, pages 101–180, 1963.

واژه‌نامه

الف

ت	heuristic ابتکاری
experimental تجربی	high dimensions ابعاد بالا
density تراکم	bias اریب
approximation تقریب	threshold آستانه
partition تقسیم‌بندی	pigeonhole principle اصل لانه‌ی کبوتری
mesh توری	NP-Hard سخت
distributed توزیع‌شده	transition انتقال

ب

ث	online برخط
invariant ثابت حلقه	linear programming برنامه‌ریزی خطی
ج	optimum بهینه
separable جداپذیر	maximum بیشینه

پ

جعبه سیاه	outlier پرت
جوینار داده	query پرسمان
پ	cover پوشش
پ	complexity پیچیدگی

pseudocode	شبه کد	extreme	حدی
object	شیء	greedy	حریصانه

ص

satisfiability	صدق‌پذیری
--------------------------	-----------

ض

factor	ضریب
------------------	------

ع

width	عرض
-----------------	-----

غ

dominate	غلبه
--------------------	------

ف

distance	فاصله
--------------------	-------

space	فضا
-----------------	-----

ق

deterministic	قطعی
-------------------------	------

ک

efficient	کارا
---------------------	------

candidate	کاندیدا
---------------------	---------

minimum	کمینه
-------------------	-------

خ

cluster	خوشه
-------------------	------

linear	خطی
------------------	-----

د

data	داده
----------------	------

data mining	داده‌کاوی
-----------------------	-----------

outlier data	داده‌ی پرت
------------------------	------------

doubling	دو برابر سازی
--------------------	---------------

binary	دودویی
------------------	--------

ر

vertex	رأس
------------------	-----

formal	رسمی
------------------	------

ز

sublinear	زیرخطی
---------------------	--------

س

amortized	سرشکن
---------------------	-------

hierarchichal	سلسه‌مراتبی
-------------------------	-------------

ش

inversion نابه‌جایی

invariant ناورد

point نقطه

center point نقطه‌ی مرکزی

half space نیم‌فضا

گ

pass گذر

م

set مجموعه

coreset مجموعه هسته

planar مسطح

parallelization موازی‌سازی

buffer میان‌گیر

ه

kernal هسته

ی

edge یال

ن

Abstract

The k -center problem—covering a set of points using k congruent balls with minimum radius—is a well-known clustering model in computer science with a wide range of applications. The k -center is a well known NP-Hard problem. In this thesis, we focus on the k -center problem with outliers in high dimensional data streams. Due to increase in data size, we focus on the data stream model of the problem. Moreover, in real-world applications, where input points are noisy, it is very important to consider outliers.

In this thesis, we study 1-center and 2-center with outliers in high dimensional data streams in Euclidean space. We provide a 1.7-approximation streaming algorithm for 1-center with z outliers (for constant z), which improves previous 1.73-approximation algorithm. We also provide a $(1.8 + \epsilon)$ -approximation streaming algorithm for 2-center problem with outliers, improving upon the previous $(4 + \epsilon)$ -approximation algorithm available for the problem. The space complexity and update time of both algorithms are $\text{poly}(z, d, \frac{1}{\epsilon})$, independent of the size of the stream.

Keywords: Clustering, k -Center, Streaming Data, Approximation Algorithm



Sharif University of Technology

Department of Computer Engineering

M.Sc. Thesis

Approximation Algorithms for Clustering Points in the Data Stream Model

By:

Behnam Hatami-Varzaneh

Supervisor:

Dr. Hamid Zarrabi-Zadeh

September 2015