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

بسم الله الرحمن الرحیم

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

این قسمت باید تکمیل گردد.

کلیدواژه‌ها: خوشه‌بندی، kــ مرکز، جویبار داده، الگوریتم تقریبی

فهرست مطالب

فهرست شکل‌ها

فهرست جدول‌ها

فصل ۱

مقدمه

مسئله‌ی **خوشه‌بندی**[1] یکی از مهم‌ترین مسائل داده‌کاوی[2] به‌حساب می‌آید. در این مسئله هدف، دسته‌بندی تعدادی جسم به‌گونه‌ای است که اجسام 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[3]، پردازش تصویر[4]، بازیابی اطلاعات[5] و بایوانفورماتیک[6] مورد استفاده قرار می‌گیرد [۲].

مسئله‌ی خوشه‌بندی، به‌خودی‌خود یک مسئله‌ی الگوریتمی به حساب نمی‌آید. راه‌حل‌های الگوریتمی بسیار زیادی برای خوشه‌بندی تعریف شده است. این الگوریتم‌ها را براساس رویکردهای مختلفی که به مسئله دارند، می‌توان در یکی از چهار دسته‌بندی زیر قرار داد:

- خوشه‌بندی‌های سلسه‌مراتبی[7]
- خوشه‌بندی‌های مرکزگرا[8]

[1] Clustering
[2] Data mining
[3] Pattern recognition
[4] Image analysis
[5] Information retrieval
[6] Bioinformatics
[7] Hierarchical clustering
[8] Centroid-based clustering

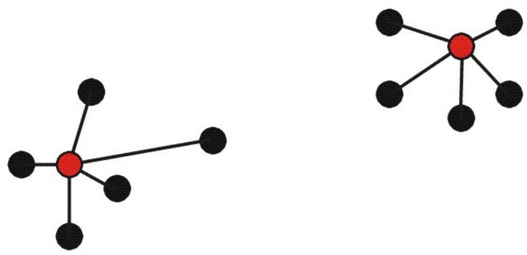
- خوشه‌بندی‌های مبتنی بر توزیع[۹] نقاط
- خوشه‌بندی‌های مبتنی بر چگالی[۱۰] نقاط

در عمل هیچ‌کدام از راه‌حل‌های بالا بر دیگری ارجحیت ندارند 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 راه‌حل‌های شناخته‌شده برای مسئله‌ی خوشه‌بندی، الگوریتم **kــ مرکز** است. در این الگوریتم هدف، پیدا کردن k نقطه به عنوان مرکز دسته‌ها است به‌طوری‌که شعاع دسته‌ها تا حد ممکن کمینه شود. در نظریه‌ی گراف، مسئله‌ی kــ مرکز متریک[۱۱] یا مسئله‌ی **استقرار تجهیزات متریک**[۱۲] یک مسئله‌ی **بهینه‌سازی ترکیبیاتی**[۱۳] است.

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[۱۴] باشند و یا به زبان دیگر، یک گراف کامل داشته باشیم که فاصله‌ها در آن در رابطه‌ی مثلثی[۱۵] صدق می‌کنند. مثالی از مسئله‌ی ۲ــ مرکز در شکل ۱ــ۱ نشان داده شده است.

در این پژوهش، مسئله‌ی kــ مرکز با متریک‌های خاص و برای kهای کوچک مورد بررسی قرار گرفته است و هر کدام از جنبه‌های متفاوتی بهبود یافته است. در بخش بعدی، تعریف رسمی[۱۶] از مسائلی که در این پایان‌نامه مورد بررسی قرار می‌گیرند را بیان نموده و در مورد هر کدام توضیح مختصری می‌دهیم.

[۹] Distribution-based
[۱۰] Density-based
[۱۱] Metric
[۱۲] Metric facility location
[۱۳] Combinatorial optimization
[۱۴] Metric space
[۱۵] Triangle equation
[۱۶] Formal



شکل ۱ــ۱: نمونه‌ای ازمسئله‌ی ۲ــ مرکز

۱ــ۱ تعریف مسئله

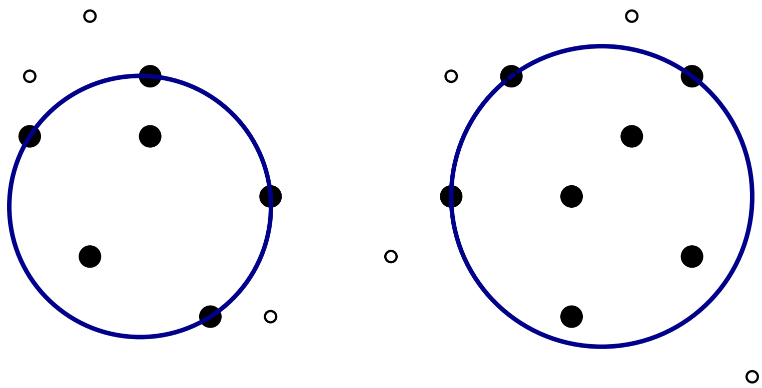
تعریف دقیق‌تر مسئله‌ی kــ مرکز در زیر آمده است:

مسئله‌ی ۱ــ۱ (kــ مرکز) *یک گراف کامل بدون جهت $G = (V, E)$ با تابع فاصله‌ی d، که از نامساوی مثلثی پیروی می‌کند داده‌شده است. زیرمجموعه‌ی $S \subseteq V$ با اندازه‌ی k را به‌گونه‌ای انتخاب کنید به‌طوری‌که عبارت زیر را کمینه کند:*

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

گونه‌های مختلفی از مسئله‌ی kــ مرکز با محدودیت‌های متفاوتی به‌وسیله‌ی 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 بپردازیم. 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 به‌عنوان 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 مسئله‌ی kــ مرکز تبدیل می‌شود. نمونه‌ای از مسئله‌ی ۲ــ مرکز با ۷ داده‌ی پرت را در شکل ۱ــ۲ می‌توانید ببینید. تعریف دقیق‌تر این مسئله در زیر آمده است:

مسئله‌ی ۱ــ۲ (kــ مرکز با داده‌های پرت) *یک گراف کامل بدون جهت $G = (V, E)$ با تابع فاصله‌ی*



شکل ۱ـ۲: نمونه‌ای ازمسئله‌ی ۲ـ مرکز با داده‌های پرت

d، که از نامساوی مثلثی پیروی می‌کند داده‌شده است. زیرمجموعه‌ی $Z \subseteq V$ با اندازه‌ی z و مجموعه‌ی $S \subseteq V - Z$ با اندازه‌ی k را انتخاب کنید به‌طوری‌که عبارت زیر را کمینه کند:

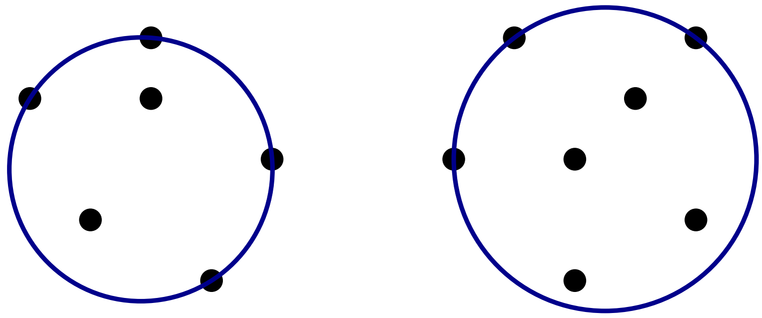
$$\max_{v \in (V-Z)} \{\min_{s \in S} d(v,s)\}$$

گونه‌ی دیگری از مسئله‌ی kـ مرکز که در سال‌های اخیر مورد توجه قرار گرفته است، حالت جویبار داده‌ی آن است. در این‌گونه از مسئله‌ی kـ مرکز، در ابتدا تمام نقاط در دسترس نیستند، بلکه به‌مرور زمان نقاط در دسترس قرار می‌گیرند. محدودیت دومی که وجود دارد، محدودیت شدید حافظه است، به‌طوری‌که نمی‌توان تمام نقاط را در حافظه نگه داشت و بعضاً حتی امکان نگه‌داری در حافظه‌ی جانبی نیز وجود ندارد و به‌طور معمول باید مرتبه‌ی حافظه‌ای کم‌تر از مرتبه حافظه‌ی **خطی**[۱۷] متناسب با تعداد نقاط استفاده نمود. از این به بعد به چنین مرتبه‌ای، مرتبه‌ی **زیرخطی** می‌گوییم. مدلی که ما در این پژوهش بر روی آن تمرکز داریم مدل جویبار داده تک‌گذره [۳] 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 kـ 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. این‌گونه از مسئله‌ی kـ مرکز، معمولاً تنها برای

[۱۷] Sublinear

L_pـ 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 ۲ـ مرکز در حالت پیوسته، در شکل ۱ـ۳ نشان داده شده است. تعریف دقیق گونه‌ی جویبار داده‌ی مسئله‌ی kـ مرکز، در زیر آمده است:



شکل ۱ـ۳: نمونه‌ای ازمسئله‌ی ۲ـ مرکز در حالت پیوسته

مسئله‌ی ۱ـ۳ (kـ 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 به‌طوری‌که عبارت زیر را کمینه شود:*

$$\max_{u \in U}\{\min_{s \in S} L_p(u, s)\}$$

از آنجایی که گونه‌ی جویبار داده و داده پرت مسئله‌ی kـ مرکز به علت داغ شدن مبحث داده‌های بزرگ[۱۸]، به تازگی مورد توجه قرار گرفته است و نتایج به‌دست‌آمده قابل بهبود است. در این تحقیق سعی شده است که تمرکز بر روی این‌گونه‌ی خاص از مسئله باشد. همچنین در این پژوهش سعی می‌شود گونه‌های مسئله را برای انواع متریک‌ها و برای kهای کوچک نیز مورد بررسی قرار داد.

۱ـ۲ اهمیت موضوع

مسئله‌ی kـ 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 kـ مرکز در بعضی مسائل مانند ارسال نامه از تعدادی مراکز پستی به گیرنده‌ها، نیاز دارد که

[۱۸] Big data

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 به‌صرفه است که نقاط پرت در نظر گرفته نشود. به طور مثال، Kmarts در سال ۲۰۱۰ اعلام کرده است که به ۸۸ درصد جمعیت آمریکا با فاصله‌ای حداکثر ۶ مایل، می‌تواند سرویس دهد. درصورتی‌که 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 به‌صرفه نبود [۴]. علاوه بر دو گونه‌ی مطرح‌شده در این قسمت، گونه‌های دیگر از مسئله‌ی kــ مرکز در مرجع [۴] آمده است.

۱ــ۳ ادبیات موضوع

همان‌طور که ذکر شد مسئله‌ی kــ مرکز در حالت داده‌های پرت و جویبار داده، گونه‌های تعمیم‌یافته از مسئله‌ی kــ مرکز هستند و در حالت‌های خاص به مسئله‌ی kــ مرکز کاهش پیدا می‌کنند. مسئله‌ی kــ مرکز در حوزه‌ی مسائل ان‌پی‌ــ سخت[۱۹] قرار می‌گیرد و با فرض $P \neq NP$ الگوریتم دقیق با زمان چندجمله‌ای برای آن وجود ندارد. بنابراین برای حل کارای[۲۰] این مسائل از الگوریتم‌های تقریبی[۲۱] استفاده می‌شود.

برای مسئله‌ی kــ مرکز، دو الگوریتم تقریبی معروف وجود دارد. در الگوریتم اول، که به روش حریصانه[۲۲] عمل می‌کند، در هر مرحله بهترین مرکز ممکن را انتخاب می‌کند به طوری تا حد ممکن از مراکز قبلی دور باشد. این الگوریتم، الگوریتم تقریبی با ضریب تقریب ۲ ارائه می‌دهد. در الگوریتم دوم، با استفاده از مسئله‌ی مجموعه‌ی غالب کمینه[۲۳]، الگوریتمی با ضریب تقریب ۲ ارائه می‌گردد. همچنین ثابت شده است، که بهتر از این ضریب تقریب، الگوریتمی نمی‌توان ارائه داد مگر آن‌که $P = NP$ باشد.

[۱۹] NP-hard
[۲۰] Efficient
[۲۱] Approximation algorithm
[۲۲] Greedy
[۲۳] Dominating set

۱ـ۴ اهداف تحقیق

در این پایان‌نامه مسئله‌ی kـ مرکز در حالت جویبار داده با داده‌های پرت در حالت‌های مختلف مورد بررسی قرار می‌گیرد و سعی خواهد شد که نتایج قبلی در این مسائل را از جنبه‌های مختلفی مورد بهبود دهد.

اولین مسئله‌ای که مورد بررسی قرار گرفته است، ارائه الگوریتمی تقریبی، برای مسئله‌ی ۱ ـ مرکز در حالت جویبار داده است به‌طوری‌که، نه تنها تمام نقاط ورودی را می‌پوشاند بلکه تضمین می‌کند که دایره‌ی بهینه‌ی جواب ۱ ـ مرکز برای نقاط ورودی را نیز بپوشاند. در تلاش اول، الگوریتم جدیدی با حافظه و زمان به‌روزرسانی $\mathcal{O}(\frac{d}{\epsilon})$ و با ضریب تقریب $\epsilon+$۱/۸ ، برای این مسئله ارائه می‌گردد. با بررسی‌های بیش‌تر، الگوریتم دیگری با ضریب تقریب ۱/۷ ، با الگوریتمی کاملاً متفاوت، با حافظه‌ی $\mathcal{O}(d)$ برای این مسئله ارائه می‌گردد.

مسئله‌ی دومی که مورد بررسی قرار می‌گیرد، مسئله‌ی ۱ ـ مرکز در حالت جویبار داده با داده‌های پرت است. در ابتدا الگوریتمی ساده با حافظه‌ی $\mathcal{O}(zd)$ و زمان به‌روزرسانی $\mathcal{O}(d+\log(z))$ با ضریب تقریب ۲ برای این مسئله ارائه می‌گردد. در بررسی‌های بعدی، با استفاده از نتایج به‌دست آمده در قسمت قبل، برای z های کوچک، الگوریتمی با حافظه‌ی $\mathcal{O}(dz^{d+1})$ با ضریب تقریب ۱/۷ برای این مسئله ارائه شد که ضریب تقریب بهترین الگوریتم موجود را، از ۱/۷۳ به ۱/۷ کاهش می‌دهد.

مسئله‌ی سومی که مورد بررسی قرار گرفت، مسئله‌ی ۲ ـ مرکز در حالت جویبار داده با داده‌های پرت است. بهترین الگوریتمی که در حال حاضر برای این مسئله وجود دارد، یک الگوریتم با ضریب تقریب $\epsilon +$ ۴ است [۵]. ما با ارائه‌ی الگوریتمی جدید، الگوریتمی با ضریب تقریب $\epsilon +$ ۱/۸ برای این مسئله ارائه دادیم که بهبودی قابل توجه محسوب می‌شود.

۱ـ۵ ساختار پایان‌نامه

این پایان‌نامه در پنج فصل به شرح زیر ارائه خواهد شد. در فصل دوم به بیان مفاهیم و تعاریف مرتبط با موضوعات مورد بررسی در بخش‌های دیگر خواهیم پرداخت. سعی شده، تا حد امکان با زبان ساده و ارجاع‌های مناسب، پایه‌ی لازم را برای فصول بعدی در این فصل فراهم آوریم. فصل سوم این پایان‌نامه شامل مطالعه و بررسی کارهای پیشین انجام شده مرتبط با موضوع این پایان‌نامه خواهد بود. این فصل

در سه بخش تنظیم گردیده است. در بخش اول، مسئله‌ی kـ مرکز در حالت ایستا مورد بررسی قرار می‌گیرد. در بخش دوم، حالت جویبار داده‌ی مسئله و مجموعه هسته‌های[۲۴] مطرح برای این مسئله مورد بررسی قرار می‌گیرد. در نهایت، در بخش سوم، مسئله‌ی kـ مرکز با داده‌های پرت مورد بررسی قرار می‌گیرد.

در فصل چهارم، نتایج جدیدی که در این پایان‌نامه به‌دست آمده است، ارائه می‌شود. این نتایج شامل الگوریتم‌های جدید برای مسئله‌ی ۱ ـ مرکز در حالت جویبار داده، مسئله‌ی ۱ ـ مرکز با داده‌های پرت در حالت جویبار داده و مسئله‌ی دو مرکز در حالت جویبار داده با داده‌های پرت می‌شود.

در فصل پنجم، به جمع‌بندی کارهای انجام شده در این پژوهش و ارائه‌ی پیشنهادهایی برای انجام کارهای آتی و تعمیم‌هایی که از راه‌حل ارائه شده وجود دارد، خواهیم پرداخت.

[۲۴] 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 سه بخش، مسائل ان‌پی‌ـ‌سخت، الگوریتم‌های تقریبی و الگوریتم‌های جویبار داده تقسیم می‌شود.

۲ـ۱ مسائل ان‌پی‌ـ‌سخت

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 به‌وسیله‌ی تورینگ[۳] ارائه شد، این مسئله معادل پیدا کردن الگوریتمی برای این مسئله است که اثبات می‌شود امکان‌پذیر نیست [۶]. 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

[۱] Hilbert
[۲] Integral root
[۳] Touring

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 نسبتا 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 تصمیم پذیر بیان می‌کنند.

مسئله‌ی ۲ـ۱ (مسائل تصمیم‌گیری)[۷] *به دسته‌ای از مسائل گفته می‌شود که پاسخ آن‌ها تنها بله یا خیر است.*

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

مسئله‌ی ۲ـ۲ (نسخه‌ی تصمیم پذیر ۱ـ 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 ۱ـ 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 [۶] مراجعه نمود.

---

[۴] Efficiency

[۵] Complexity theory

[۶] Formal

[۷] Decision problems

[۸] Verifiable

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 درصورتی‌که زمانی، خلاف این فرض اثبات گردد، آنگاه قسمت عمده‌ای از علوم کامپیوتر زیر سؤال می‌رود.

در نظریه‌ی پیچیدگی، برای دسته‌بندی مسائل، یکی از روش‌های دسته‌بندی کاهش چندجمله‌ای[۱۰] مسائل به یکدیگر است.

تعریف ۲ـ۱ می‌گوییم مسئله‌ی 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 جستجوهایی 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 ان‌پی‌ـ 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 کوک‌ـ لوین** ثابت کردند مسئله‌ی صدق‌پذیری[۱۱] یک مسئله‌ی ان‌پی‌ـ سخت است [۶]. با پایه قرار دادن این اثبات و استفاده از تکنیک کاهش، اثبات ان‌پی‌ـ سخت بودن سایر مسائل، بسیار ساده‌تر گردید. در ادامه مسئله‌ی پوشش رأسی[۱۲] را تعریف می‌کنیم.

۲ـ۱ـ۱ پوشش رأسی

در این پایان‌نامه، از این مسئله به عنوان مسئله‌ی پایه برای اثبات ان‌پی‌ـ سخت بودن مسئله‌ی kـ مرکز استفاده می‌شود. تعریف این مسئله مطابق زیر است:

[۹] Open problem
[۱۰] Polynomial Reduction
[۱۱] Satisfiability problem
[۱۲] Vertex Coverage

مسئله‌ی ۲ـ۳ (پوشش رأسی) *گراف بدون جهت $G(V,E)$ داده شده است. هدف مسئله پیدا کردن مجموعه‌ی $S \subset V$ با کم‌ترین تعداد اعضا است به‌طوری‌که هر رأس $v \in V$ در یکی از شرایط زیر صدق کند:*

- $v \in S$
- *وجود دارد رأسی* $u \in S$ *به طوری* $(v,u) \in E$

به عبارت ساده‌تر هر رأسی یا خودش یا یکی از همسایگانش داخل مجموعه‌ی S قرار داشته باشد.

نسخه‌ی تصمیم‌گیری این مسئله به این‌گونه تعریف می‌شود که آیا گراف داده‌شده دارای پوشش رأسی با اندازه‌ی k است یا نه.

قضیه‌ی ۲ـ۱ *مسئله‌ی پوشش رأسی، یک مسئله‌ی ان‌پی‌ـ سخت است.*

اثبات. *برای مشاهده‌ی اثبات ان‌پی‌ـ سخت بودن مسئله‌ی پوشش رأسی، نیاز به زنجیره‌ای از مسائل که از مسئله‌ی صدق‌پذیری شروع می‌شود است، به‌طوری‌که هر عضو از این زنجیره، به عضو بعدی در زمان چندجمله‌ای کاهش می‌یابد و در نهایت نتیجه می‌شود که مسئله‌ی صدق‌پذیری در زمان چندجمله‌ای به مسئله‌ی پوشش رأسی کاهش می‌یابد و در نتیجه چون مسئله‌ی صدق‌پذیری یک مسئله‌ی ان‌پی‌ـ سخت است. بنابراین مسئله‌ی پوشش رأسی نیز ان‌پی‌ـ سخت خواهد بود. برای مطالعه‌ی روند اثبات به مرجع [۶] مراجعه کنید.*

□

۲ـ۲ 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 بیش از ۳، یک الگوریتم ناکارآمد محسوب می‌شود. 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 هستند و در نتیجه راه‌حل چندجمله‌ای ندارند، یا اگر راه‌حل چند جمله‌ای 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 مکاشفه‌ای[۱۳] گوناگونی برای مسائل مختلف به خصوص مسائل ان‌پی 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 مکاشفه‌ای، عدم امکان استفاده از آن‌ها برای تمام کاربردها است. بنابراین در رویکرد دوم که اخیراً نیز مطرح شده است و عمر کمی دارد، سعی در ارائه‌ی الگوریتم‌های مکاشفه‌ای شده است که تضمین می‌کنند اختلاف زیادی با الگوریتمی که جواب بهینه می‌دهد، نداشته باشند. در واقع این الگوریتم‌ها همواره و در هر شرایطی، تقریبی از جواب بهینه را ارائه می‌دهند. به چنین الگوریتم‌هایی، ***الگوریتم‌های تقریبی***[۱۴] *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[۱۵] *دارند و به همین دلیل برای داده‌های حجیم بسیار کاربرد دارند.*

[۱۳] Heuristic

[۱۴] Approximation Algorithm

[۱۵] Sublinear

مسئله	کران پایین تقریب‌پذیری
پوشش رأسی	۱,۳۶۰۶ [۷]
kــ مرکز	۲[۸]
kــ مرکز در فضای اقلیدسی	۱,۸۲۲[۹]
۱ــ مرکز در حالت جویبار داده	$\frac{۱+\sqrt{۲}}{۲}$ [۱۰]
kــ مرکز با نقاط پرت و نقاط اجباری	۳[۴]

جدول ۲ــ۱: نمونه‌هایی از ضرایب تقریب برای مسائل بهینه‌سازی

۲ــ۲ــ۱ میزان تقریب‌پذیری مسائل

همان‌طور که تا اینجا دیدیم، یکی آز 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. حتی 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 kــ مرکز وجود ندارد مگر اینکه $P = NP$ باشد. برای مسائل مختلف، معمولاً می‌توان کران پایینی برای میزان تقریب‌پذیری آن‌ها ارائه داد. در واقع برای مسائل ان‌پی، علاوه بر این الگوریتم کارآمدی وجود ندارد، بعضاً الگوریتم تقریبی برای حل آن با ضریبی تقریب کم و نزدیک به یک نیز وجود ندارد. در جدول ۲ــ۱ از میزان تقریبی مسائل مختلفی که در این پایان‌نامه مورد استفاده قرار می‌گیرد ببینید.

۲ــ۳ الگوریتم‌های جویبارداده

در علوم کامپیوتر، الگوریتم جویبار داده به الگوریتم‌هایی گفته می‌شود که برای پردازش جویبارهای داده طراحی شده‌اند به طوری که ورودی آن، به صورت دنباله‌ای از داده‌ها داده می‌شود و تنها می‌توان تعدادی محدود بار، از روی دنباله گذر کرد (معمولاً تنها یک بار). الگوریتم‌های جویبار داده معمولاً محدودیت شدیدی در 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 را ارائه دهد.

۲ـ۳ـ۱ مقدمه

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[۱۶] 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 قابل توزیع‌پذیر بوده و توانایی اجرا شدن به صورت چندروندی[۱۷] را دارا باشند.*

---
[۱۶] Locality
[۱۷] Multi Thread

۲ـ۳ـ۲ گونه‌های مطرح

در مدل جویبار داده، بعضی یا همه‌ی نقاط ورودی که باید پردازش گردند برای دسترسی تصادفی[۱۸] 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 [۳].

هر جویبار را می‌توان به عنوان دنباله‌ای مرتب از نقاط (یا به‌روزرسانی‌ها[۱۹]) 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 [۳].

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

- 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[۲۰] داخل دنباله. [۳]

۲ـ۳ـ۳ 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 هر داده

[۱۸] Random access
[۱۹] Update
[۲۰] Inversion

الگوریتم‌های جویبار داده تشابه‌های زیادی با الگوریتم‌های برخط[۲۱] 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.

۲ـ۳ـ۴ مجموعه هسته

یکی از تکنیک‌های رایج در الگوریتم‌های جویبار داده، نگه‌داری نماینده‌ای با اندازه‌ی بسیار کوچک‌تر نسبت به‌اندازه‌ی جویبار داده است. این مجموعه معمولاً دغدغه‌ی محدودیت حافظه را در الگوریتم‌های جویبار داده برطرف می‌کند. به چنین مجموعه‌ای، مجموعه هسته می‌گوییم. حال به تعریف رسمی مجموعه هسته می‌پردازیم:

تعریف ۲ـ۲ *فرض کنید μ یک تابع اندازه‌گیری[۲۲] (همانند تابع عرض مجموعه‌ای از نقاط) از $\mathbb{R}^d$ به اعداد حقیقی نامنفی $\mathbb{R}^+ \cup \{0\}$ 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*

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

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

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

یکی از مجموعه هسته‌های معروف، مجموعه هسته‌ی مطرح برای تابع اندازه‌گیری عرض نقاط است. به چنین مجموعه هسته‌ای به اختصار ϵـ هسته[۲۳] گفته می‌شود. تعریف دقیق ϵـ هسته در زیر آمده است:

[۲۱] Online

[۲۲] Measure function

[۲۳] ϵ-Kernel

تعریف ۲ــ۳ *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 $\langle ., . \rangle$ همان ضرب داخلی دو بردار است. برای متغیر $\epsilon > 0$، زیرمجموعه‌ی $Q \subset P$ یک ϵــ هسته نامیده می‌شود اگر به ازای هر $u \in S^{d-1}$ داشته باشیم:

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

ϵــ هسته یکی از اساسی‌ترین مجموعه هسته‌های مطرح است و برای طیف وسیعی از مسائل قابل استفاده است. الگوریتم‌های زیادی برای محاسبه‌ی ϵــ هسته در حالت ایستا ارائه شده است [۱۱].

یکی از روش‌هایی که در تبدیل یک الگوریتم از حالت ایستا به حالت جویبار داده ارائه شده است، استفاده از روش دوبرابرسازی[۲۴] است. این روش یک روش عام و قابل استفاده برای طیف وسیعی از مسائل است. به عنوان نمونه، دوبرابرسازی را بر روی ϵــ هسته مورد بررسی قرار می‌دهیم. ϵــ هسته، دارای دو خاصیت اساسی است که آن را قابل استفاده برای تکنیک دوبرابرسازی است.

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

ایده‌ی اصلی تکنیک دوبرابرسازی، 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

[۲۴] Doubling

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.

فصل ۳

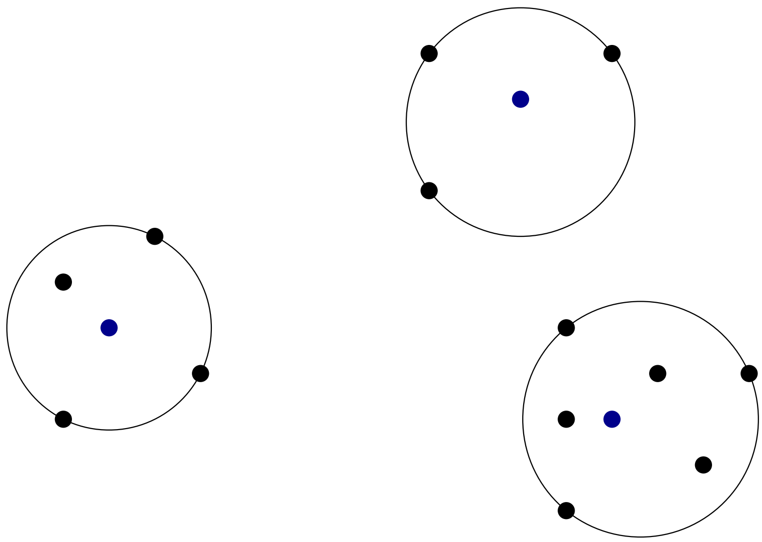
کارهای پیشین

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

۳ـ۱ kـ مرکز در حالت ایستا

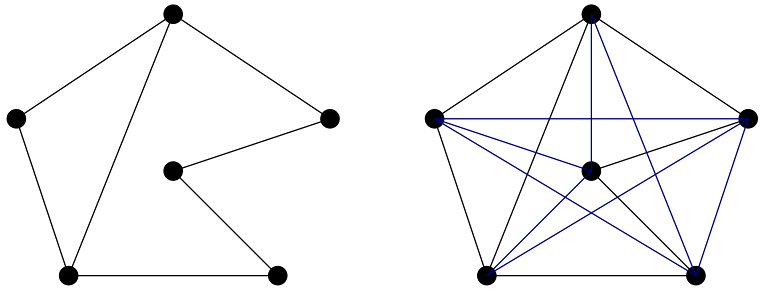
مسئله‌ی kـ مرکز به عنوان مسئله‌ی شناخته شده در علوم کامپیوتر مطرح است. این مسئله، در واقع یک مسئله‌ی بهینه‌سازی است که سعی در کاهش بیش‌ترین فاصله نقاط از مرکز دسته‌ها را دارد. سختی اصلی این مسئله در انتخاب مرکز دسته‌هاست. زیرا اگر بتوانیم مرکز دسته‌ها را به درستی تشخیص دهیم، کافی است هر نقطه را به دسته‌ای که نزدیک‌ترین مرکز را دارد، تخصیص دهیم. به وضوح چنین تخصیصی بهینه‌ترین تخصیص ممکن است. نمونه‌ای از این تخصیص را در شکل ۳ـ۱ نشان داده شده است.

در سال ۱۹۷۹ ، نه تنها اثبات گردید که این مسئله در حالت کلی یک مسئله‌ی ان‌پی‌ـ سخت است [۱۲]، بلکه ثابت شده است که این مسئله در صفحه‌ی دو بعدی و با متریک اقلیدسی نیز ان‌پی‌ـ سخت است [۱۳]. فراتر از این، ثابت شده است که برای مسئله‌ی kـ مرکز با متریک دلخواه هیچ الگوریتم تقریبی با ضریب تقریب بهتر از ۲ وجود ندارد. ایده‌ی اصلی این کران پایین، کاهش مسئله‌ی پوشش رأسی، به مسئله‌ی kـ مرکز است. برای چنین کاهشی کافی است، از روی گراف اصلی که می‌خواهیم



شکل ۳ـ۱: نمونه‌ای از تخصیص نقاط به ازای مراکز آبی رنگ.

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 می‌دهیم. نمونه‌ای از چنین تبدیلی را در شکل ۳ـ۲ می‌توانید مشاهده کنید. حال اگر الگوریتمی بتواند مسئله‌ی kـ مرکز را با ضریب تقریب بهتر از ۲ حل نماید، آن‌گاه گراف جدید دارای یک kـ 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 kـ مرکز، الگوریتم تقریبی با ضریب تقریب بهتر از ۱/۸۲۲ وجود ندارد [۹].



شکل ۳ـ۲: نمونه‌ای از تبدیل یک گراف ورودی مسئله‌ی پوشش رأسی به یک ورودی مسئله‌ی kـ مرکز(در گراف سمت چپ، یال‌های سیاه وزن ۱ و یال‌های آبی، وزن ۲ دارند)

[۱] Euclidean space

الگوریتم ۱ الگوریتم گنزالز

S را برابر مجموعه تهی قرار بده.
عنصر دلخواه از مجموعه نقاط V را به S اضافه کن.
به ازای i بین ۲ تا k:
  v را نقطه‌ای از V در نظر بگیرید که بیش‌ترین فاصله را از مجموعه‌ی S دارد.
  v را S اضافه کن.
S را برگردان

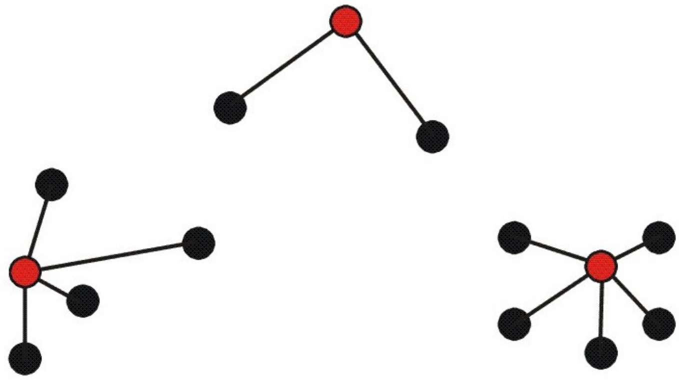
یکی از اولین الگوریتم‌های تقریبی برای مسئله‌ی kــ مرکز به وسیله‌ی گنزالز[۲] *ارائه شده است* [۱۴]. *این الگوریتم یک الگوریتم تقریبی با ضریب ۲ است و در زمان $\mathcal{O}(kn)$ قابل اجراست. الگوریتم گنزالز، از نظر روش برخورد با مسئله، یک الگوریتم حریصانه*[۳] *محسوب می‌شود. برای این‌که عملکرد الگوریتم گنزالز را درک کنید، نیاز به تعریف فاصله‌ی یک نقطه از یک مجموعه نقطه داریم.*

تعریف ۳ــ ۱ *فاصله‌ی نقطه‌ی v از مجموعه‌ای ناتهی از نقاط S را برابر فاصله‌ی نقطه‌ای درون S از v تعریف می‌کنیم، به گونه‌ای 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 موجود را به عنوان مرکز دسته‌ی جدید به مجموعه مراکز دسته‌ها اضافه می‌کند. با اجرای الگوریتم تا k مرحله، مراکز دسته‌ها انتخاب می‌شود. 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 kــ مرکز است. فدر[۴] *و سایرین، زمان اجرای الگوریتم گنزالز را برای هر L_pــ متریک به مرتبه‌ی $\mathcal{O}(n \log k)$ بهبود بخشیدند. نمونه‌ای از اجرای الگوریتم گنزالز، در شکل ۳ــ ۳ نشان داده شده است.*

[۲] Gonzalez
[۳] Greedy
[۴] Feder



شکل ۳ـ۳: نمونه‌ای از حل مسئله‌ی ۳ـ مرکز با الگوریتم گنزالز

تا به اینجا، تنها بر روی حالت کلی مسئله‌ی kـ 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 ـ مرکز، که k تعداد دسته‌ها یا d ابعاد فضا ثابت باشند می‌پردازیم. آگاروال[۵] و سایرین الگوریتمی دقیق با زمان اجرای $n^{\mathcal{O}(k^{1-\frac{1}{d}})}$ برای مسئله kـ مرکز در فضای L_pـ متریک با ابعاد ثابت d ارائه داده‌اند [۱۵]. قابل توجه است که اگر d ثابت نباشد، مسئله‌ی kـ مرکز حتی برای متریک اقلیدسی(L_2ـ متریک) با تعداد دسته‌ی ثابت $k \geqslant 2$، ان‌پی‌ـ سخت است [۱۶].

علاوه بر حالاتی که ابعاد فضا یا تعداد دسته‌ها ثابت‌اند، مسئله‌ی kـ 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 ـ مرکز در فضای اقلیدسی با ابعاد ثابت، الگوریتم خطی با زمان اجرای $\mathcal{O}((d+1)!n)$ وجود دارد [۱۷]. 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 $\mathcal{O}(\frac{d}{n})$ است که با توجه به ثابت بودن d این احتمال برای nهای بزرگ کوچک محسوب می‌شود و زمان اجرای الگوریتم، به طور میانگین خطی خواهد بود.

در صفحه‌ی اقلیدسی برای مسئله‌ی ۲ ـ مرکز، بهترین الگوریتم را چن[۶] با زمان اجرای $\mathcal{O}(c\log^2 n\log^2 \log n)$ و حافظه‌ی $\mathcal{O}(n)$ ارائه داده است [۱۸]. برای فضای سه‌بعدی اقلیدسی نیز آگاروال و سایرین، الگوریتمی

[۵] Agarwal

[۶] Chan

با متوسط زمان اجرای $\mathcal{O}(n^{3}\log^{8} n)$ ارائه داده است [۱۹].

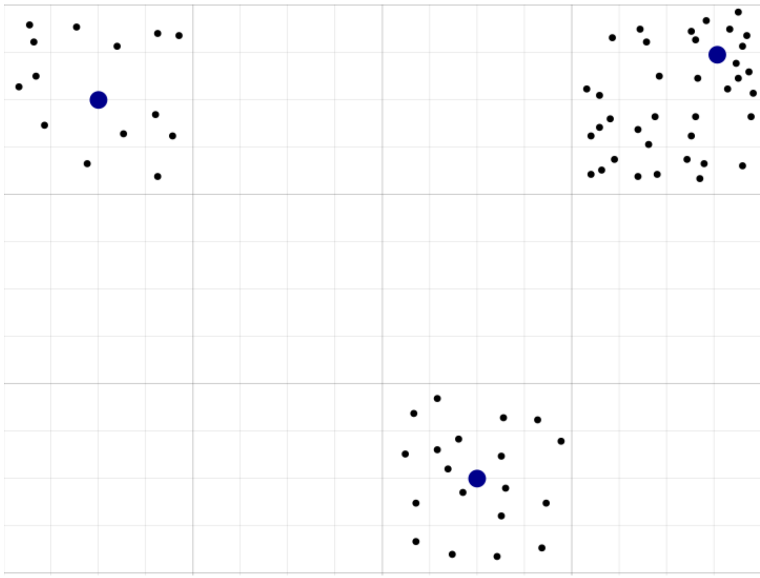
۳ـ۲ kـ مرکز در حالت جویبار داده

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

بهترین مجموعه هسته‌ای که برای مسئله‌ی kـ مرکز ارائه شده است، روش ارائه‌شده به وسیله‌ی ضرابی‌زاده برای نگه‌داری یک ϵـ هسته با حافظه‌ی $\mathcal{O}(\frac{k}{\epsilon^{d}})$ برای L_pـ متریک‌ها است [۲۰]. 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 دو تقریب از شعاع بهینه به علاوه‌ی مرکز دسته‌های پیدا شده را به دست می‌آورد. حال کافی است که با طول شعاع الگوریتم تقریبی، حول هر مرکز به دست آمده، یک توری با $\mathcal{O}(\frac{1}{\epsilon})$ شبکه‌بندی در هر بعد تشکیل داد 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 ما نیازی به نگهداری تمام نقاط ورودی نداشته و تنها نیاز به نگهداری نقاط شبکه‌بندی توری داریم. با این روش می‌توان به یک ϵـ هسته برای مسئله‌ی kـ مرکز رسید. نکته‌ی اساسی برای سازگار سازی روش ارائه‌شده با مدل جویبار داده‌ی تک‌گذره استفاده از روش دوبرارسازی[۷] رایج در الگوریتم‌های جویبار داده است. نمونه‌ای از اجرای الگوریتم ضرابی‌زاده را در شکل ۳ـ۴ نشان داده شده است. برای دیدن اثبات‌ها و توضیح بیش‌تر در مورد روش ارائه شده می‌توانید به مرجع [۲۰] مراجعه کنید.

از جمله مشکلات وارده به الگوریتم ضرابی زاده، وابستگی اندازه‌ی مجموعه‌ی هسته به ابعاد فضا است. بنابراین هسته‌ی ارائه شده به وسیله‌ی ضرابی‌زاده 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

[۷] Doubling



شکل ۳ـ۴: نمونه‌ای از توری‌بندی الگوریتم ضرابی‌زاده (نقاط آبی، مراکز به دست آمده از الگوریتم تقریبی است). پس از توری‌بندی کافی است برای هر کدام از خانه‌های شبکه‌بندی، تنها یکی را به عنوان نماینده در نظر بگیریم.

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 [۲۱]. پس از آن، گوها[۸]، به طور موازی با مک‌کاتن[۹] و سایرین، الگوریتمی با ضریب تقریب $(2+\epsilon)$ با حافظه‌ی $\mathcal{O}(\frac{dk}{\epsilon}\log\frac{1}{\epsilon})$ برای مسئله‌ی kـ مرکز در هر فضای متریکی ارائه دادند [۵، ۲۲]. در سال ۲۰۱۴، اهن[۱۰] و سایرین، الگوریتمی با همین ضریب تقریب و حافظه‌ی $\mathcal{O}((k+3)!2^k\frac{d}{\epsilon})$ ارائه داده‌اند که برای kهای ثابت، حافظه را از مرتبه‌ی $\mathcal{O}(\log\frac{1}{\epsilon})$ کاهش می‌دهد [۲۳].

تا به این‌جا ما به بررسی مسئله‌ی kـ مرکز در حالت جویبار داده بدون محدودیت خاصی پرداختیم. برای حالت‌های خاص k، به خصوص ۱ و ۲، مسئله‌ی kـ مرکز با متریک اقلیدسی مورد توجه زیادی قرار گرفته است و راه‌حل‌های بهینه‌تری نسبت به حالت کلی برای آن‌ها پیشنهاد شده است. به طور مثال،

[۸] Guha
[۹] McCutchen
[۱۰] Ahn

می‌توان یک هسته با اندازه‌ی $\mathcal{O}(\frac{1}{\epsilon^{\frac{d-1}{2}}})$، با استفاده از نقاط حدی[۱۱] *در تعداد مناسبی جهت، به صورت جویبار داده ساخت.*

الگوریتم ۲ الگوریتم ضرابی‌زاده

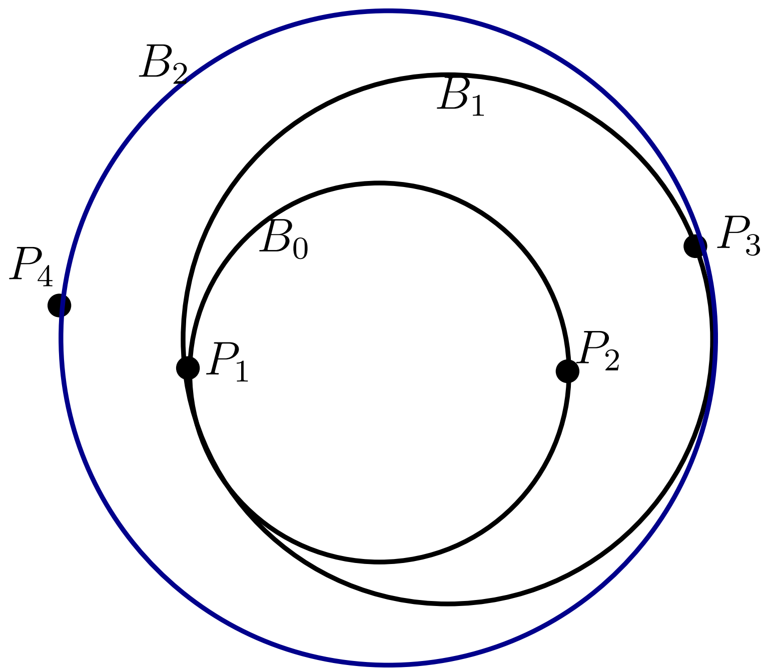
B را توپی به مرکز نقطه‌ی اول و شعاع صفر قرار دهید.
به ازای هر نقطه‌ی u در جویبار داده:
  اگر u داخل B قرار می‌گیرد:
    ادامه بده.
  در غیر این صورت:
    B را با کوچک‌ترین توپ ممکن که هردوی B و u را می‌پوشاند، جایگزین کن.
مجموعه‌ی S را برگردان.

همان‌طور که برای الگوریتم ضرابی‌زاده مطرح شد، مشکل عمده‌ی مجموعه هسته‌ی ارائه‌شده، وابستگی حافظه‌ی مصرفی آن به d است. در راستای حل این مشکل، ضرابی‌زاده و سایرین [۲۴] برای ابعاد بالا و متریک اقلیدسی، الگوریتمی با ضریب تقریب ۱/۵ و حافظه‌ی مصرفی $\mathcal{O}(d)$ ارائه دادند. در واقع در الگوریتم آن‌ها، در هر لحظه تنها یک مرکز و یک شعاع را نگه داشته می‌شود، که کم‌ترین حافظه‌ی ممکن برای مسئله‌ی ۱ـ 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 ۱/۵ـ برابر شعاع کره‌ی بهینه است. نکته‌ی قابل توجه در مورد این الگوریتم، نگه‌داری کم‌ترین حافظه‌ی ممکن برای مسئله‌ی ۱ـ مرکز است. زیرا تنها یک مرکز و یک شعاع نگه می‌دارد. نمونه‌ای از اجرای الگوریتم را برروی چهار نقطه می‌توان در شکل ۳ـ۵ دید. برای اثبات کامل‌تر می‌توانید به مرجع [۲۴] مراجعه کنید.

---

[۱۱] Extreme points



شکل ۳ـ۵: نمونه‌ای از اجرای الگوریتم ضرابی‌زاده بر روی چهار نقطه $P_1 \cdots P_4$ که به ترتیب اندیس در جویبار داده فرا می‌رسند و دایره‌های $B_0 \cdots B_2$ دایره‌هایی که الگوریتم به ترتیب نگه می‌دارد.

در ادامه، آگاروال و سایرین [۱۰] الگوریتمی تقریبی با حافظه‌ی مصرفی $O(d)$ ارائه دادند. در الگوریتم ارائه شده، ضریب تقریب، برابر $\frac{1+\sqrt{3}}{2}$ تخمین زده شد، اما با تحلیل دقیق‌تری که چن و سایرین [۲۵] بر روی همان الگوریتم انجام دادند، مشخص شد همان الگوریتم دارای ضریب تقریب ۱/۲۲ است.

الگوریتم آگاروال از الگوریتم کلارکسون و سایرین [۲۶] به عنوان یک زیرالگوریتم استفاده می‌کند. الگوریتم کلارکسون، الگوریتمی کاملا 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. ازین به بعد در هر مرحله، نقطه‌ای که از نقاط انتخاب شده‌ی قبلی بیش‌ترین فاصله را دارد به عنوان نقطه‌ی جدید انتخاب می‌کند. اگر این الگوریتم را تا $O(\frac{1}{\epsilon})$ مرحله ادامه بدهیم، به مجموعه‌ای با اندازه‌ی $O(\frac{1}{\epsilon})$ خواهیم رسید که کلارکسون و سایرین اثبات کرده‌اند که یک ϵـ هسته برای مسئله‌ی ۱ ـ مرکز است. در واقع آن‌ها نشان‌داده‌اند یک $O(\frac{1}{\epsilon})$ـ مرکز به دست آمده برای مجموعه‌ای از نقاط، یک ϵـ هسته برای مسئله‌ی ۱ ـ مرکز برای همان مجموعه نقاط 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 $\kappa = \{K_1, \cdots, K_u\}$ نگه می‌دارد و زمانی که نقطه‌ی p_{u+1} پیدا شود که در هیچ کدام از $(1+\epsilon)Meb(K_j)$ به ازای $1 \leqslant j \leqslant u$ نباشد، الگوریتم کلارکسون را برای $\cup_{j=1}^{u} K_j \cup \{p_{u+1}\}$ اجرا نموده و مجموعه هسته‌ی به دست آمده را K_{u+1} می‌نامد. با توجه به نحوه‌ی ساخته شدن K_iها، به راحتی می‌توان نشان داد رابطه‌ی زیر برقرار است:

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

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

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

را می‌پوشاند را به عنوان جواب محاسبه کنیم. چن و سایرین ثابت کرده‌اند که کره‌ی نهایی دارای شعاع حداکثر ۱/۲۲ برابر شعاع بهینه است. برای مشاهده جزئیات بیش‌تر، به [۱۰، ۲۵] مراجع کنید.

آگاروال نه تنها الگوریتمی ارائه داد که در نهایت، ثابت شد حداکثر جوابی با ضریب تقریب ۱/۲۲ 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 $\frac{1+\sqrt{2}}{2}$ داشته باشد.

قضیه‌ی ۳ـ۱ هر الگوریتم تحت مدل جویبار داده که یک αـ تقریب برای مسئله‌ی ۱ ـ مرکز برای مجموعه‌ی S شامل n نقطه در فضای $\mathbb{R}^d$ نگه می‌دارد، برای $\alpha \leqslant \frac{1+\sqrt{2}}{2}(1-\frac{2}{d^{\frac{1}{3}}})$ 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 قضیه‌ی معروف آلیس و باب[۱۲] در نظریه انتقال اطلاعات بنا شده است. برای خواندن اثبات این قضیه می‌توانید به مرجع [۱۰] مراجعه کنید.

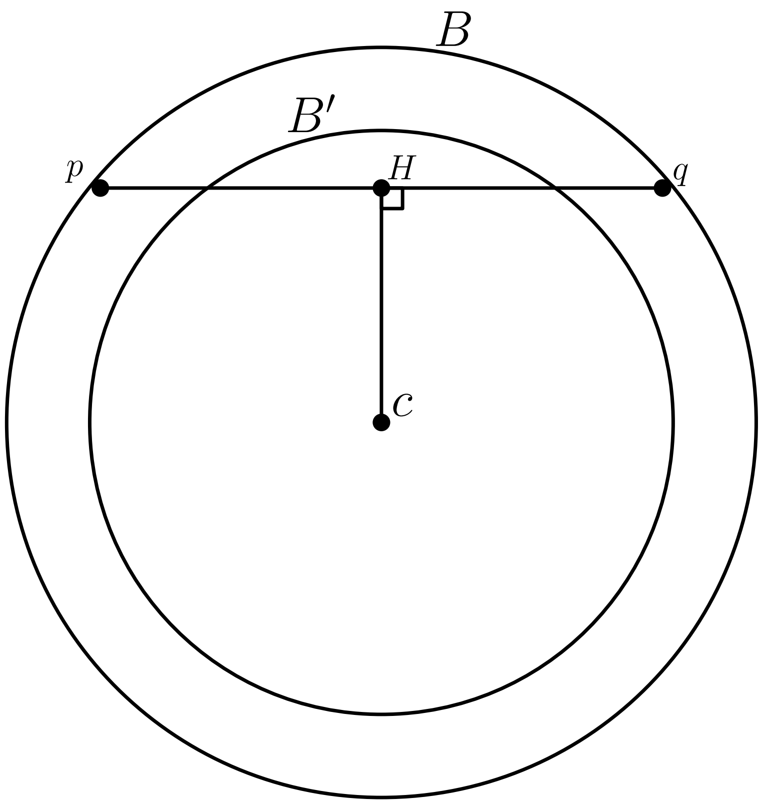
□

علاوه بر مسئله‌ی ۱ ـ مرکز، مسئله‌ی دو مرکز نیز در سال‌های اخیر مورد توجه قرار گرفته است و بهبودهایی نیز برای این مسئله ارائه شده است. آهن و سایرین [۱] در سال ۲۰۱۴، اولین الگوریم با

[۱۲] alice and bob

ضریب تقریب کم‌تر از دو را برای مسئله‌ی ۲ـ مرکز در فضای اقلیدسی ارائه دادند. این الگوریتم تقریبا پایه‌ی کار این پایان‌نامه برای حالت‌های مختلف است. به همین منظور، تعدادی از لم‌های داخل این الگوریتم که در آینده استفاده می‌شود این‌جا توضیح داده می‌شود.

لم۳ـ۲ *فرض کنید B یک کره‌ی واحد با مرکز c در فضای اقلیدسی $\mathbb{R}^d$ باشد. هر پاره‌خط pq به طول حداقل ۱/۲ که به طور کامل داخل B قرار دارد، کره‌ی $B'(c, ۰/۸)$ را قطع می‌کند.*



شکل ۳ـ۶: اثبات لم ۳ـ۲

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 ۳ـ۶ مشخص شده است، پای عمود از مرکز کره بر پاره‌خط pq را h بنامید. بدون کم شدن از کلیت مسئله فرض می‌کنیم $\|hp\| \leqslant \|hq\|$. بنابراین داریم:*

$$۰/۶ = \frac{۱/۲}{۲} \leqslant \|hq\|$$

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:

$$\|hc\| = \sqrt{\|qc\|^{۲} - \|qh\|} \leqslant \sqrt{۱ - ۰/۶^{۲}} = ۰/۸$$

بنابراین نقطه‌ی h داخل کره‌ی B' قرار می‌گیرد. از طرفی چون، $\|pq\| \leqslant ۱$ بنابراین، h داخل پاره‌خط قرار دارد و در نتیجه کره‌ی B' با پاره‌خط pq تقاطع دارد.

□

لم بالا در واقع نشان می‌دهد اگر در طول الگوریتم بتوانیم دو نقطه‌ی دور نسبت به هم (حداقل ۱/۲ برابر شعاع بهینه) از یکی از دو کره‌ی بهینه را بیابیم، این پاره‌خط از مرکز کره‌ی بهینه فاصله‌ی کمی (حداکثر ۰/۸ شعاع بهینه) دارد.

لم۳ـ۳ فرض کنید B کره‌ای به مرکز c و شعاع واحد در $\mathbb{R}^d$ باشد. پاره‌خط دلخواه pq با طول حداقل ۱/۲ که به طور کامل داخل B قرار دارد را در نظر بگیرید. هر نقطه‌ی x از پاره‌خط pq که از دو سر آن حداقل ۰/۶ فاصله داشته باشد، داخل کره‌ی $B'(c, ۰/۸)$ قرار می‌گیرد.

اثبات. اثبات این لم نیز کاملا مشابه لم ۳ـ۲ است. بدون کم شدن از کلیت مسئله همان‌طور که در شکل ۳ـ۷ مشخص شده است، فرض کنید زاویه‌ی $\angle pxc$ بزرگ‌تر مساوی ۹۰ درجه است. در نتیجه داریم:

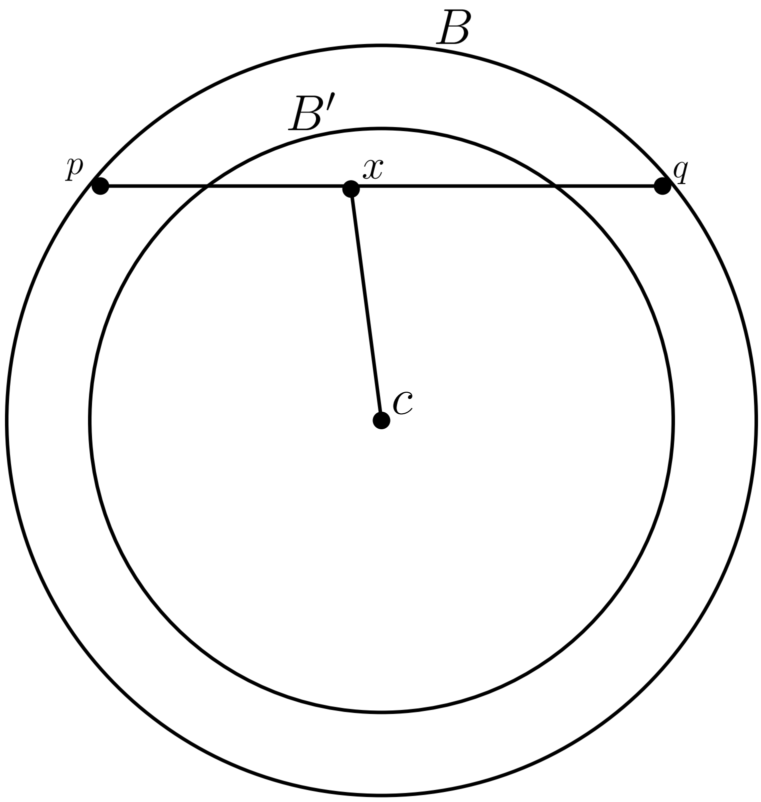
$$\sqrt{\|px\|^{۲} + \|xc\|^{۲}} \leqslant \|pc\| \leqslant ۱$$

از طرفی طبق فرض مسئله داریم:

$$۰/۶ \leqslant \|px\| \implies \|xc\| \leqslant \sqrt{۱ - ۰/۶^{۲}} = ۰/۸$$

□

الگوریتم آهن، با استفاده از دو لم بالا و تقسیم مسئله به دو حالتی که دو کره‌ی بهینه بیش از دو برابر شعاع بهینه یا کم‌تر از برابر شعاع بهینه فاصله داشته باشند، دو الگوریتم کاملا 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 بدهیم.

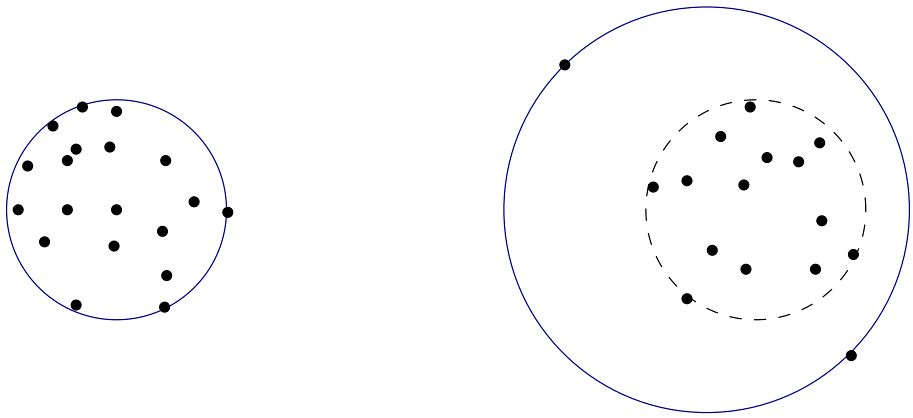


شکل ۳ـ۷: اثبات لم ۳ـ۳

۳ـ۳ kـ مرکز با داده‌های پرت

در دنیای واقعی در میان داده‌ها، داده‌های دارای اریب وجود دارد که اگر امکان تشخیص و حذف آن‌ها در حین جمع‌آوری داده‌ها وجود داشت، شعاع مسئله‌ی kـ 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 ۳ـ۸ 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 بقیه‌ی نقاط فاصله‌ی زیادی دارند از داده‌های ورودی حذف شده و سپس به عنوان ورودی مسئله‌ی kـ مرکز دسته‌های مرتب را از آن استخراج کرد.

مسئله‌ی kـ مرکز با داده‌های پرت، بسیار مشابه مسئله‌ی استقرار تجهیزات است. در مسئله‌ی استقرار تجهیزات، هدف استقرار چند مرکز ارائه دهنده‌ی خدمات است که هزینه‌ی استقرار به علاوه‌ی انتقال



شکل ۳ـ۸: کاهش قابل توجه شعاع مسئله‌ی ۲ ـ مرکز با حذف تنها دو نقطه

تجهیزات از مراکز ارائه دهنده به مکان‌ها متقاضی کمینه گردد. تعریف رسمی این مسئله در زیر آمده است:

مسئله‌ی ۳ ـ ۱ (استقرار تجهیزات) *مجموعه‌ای نقطه به عنوان مکان‌های مجاز برای استقرار تجهیزات داده شده است. هزینه‌ی استقرار تجهیزات در نقطه‌ی iاُم را برابر با f_i در نظر بگیرید. مجموعه‌ای از نقاط نیز که متقاضی تجهیزات هستند نیز داده‌شده است. به ازای هر متقاضی j و محل استقرار تجهیزات i، $d(i, j)$ برابر هزینه انتقال تجهیزات از محل استقرار به متقاضی است. مجموعه‌ای k عضوی به نام K انتخاب کنید به طوری که هزینه کلی را کمینه نماید:*

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

گونه‌های مختلفی از مسئله‌ی استقرار تجهیزات تعریف شده است. از جمله‌ی آن می‌توان به گونه‌های زیر اشاره نمود:

- *هر مرکز ارائه‌دهنده حداکثر به تعداد مشخصی از متقاضیان می‌تواند تجهیزات انتقال دهد. در واقع در این روش سعی در استقرار تجهیزات به صورت متوازن است و متناسب با قدرت ارائه‌ی تجهیزات آن‌هاست.*
- *هزینه‌ی استقرار مرکز ارائه‌دهنده متناسب با تعداد متقاضیانی که پاسخ می‌دهد است. در این روش، معمولا هر چه تعداد تقاضاهای یک مرکز بالاتر رود هزینه‌ی استقرار یا ساخت آن برای تامین چنین میزان درخواستی بالاتر می‌رود.*

- *هر متقاضی میزانی جریمه بابت عدم دریافت تقاضای خود مشخص کند. در این روش، هر متقاضی در صورت عدم دریافت تجهیزات مورد نیاز، میزانی جریمه مطالبه می‌کند و هدف کاهش مجموع هزینه‌ها به علاوه‌ی 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 تختی از محدودیت تعداد آن‌ها وجود ندارد.*

اگر به دو گونه‌ی آخر توجه بیش‌تری کنید، به شباهتشان به مسئله‌ی kـ مرکز با داده‌های پرت پی خواهید برد. می‌توان نشان داد که مسئله‌ی kـ مرکز با z داده‌ی پرت از لحاظ پیچیدگی محاسباتی هم‌ارز استقرار تجهیزات با امکان عدم پوشش z متقاضی است. همان‌طور که در زیربخش قبلی دیدیم، هیچ الگوریتمی با ضریب تقریب کم‌تر از ۲ برای مسئله‌ی kـ مرکز در حالت کلی وجود ندارد مگر این‌که $P = NP$ باشد. برای مسئله‌ی kـ مرکز با داده‌های پرت، اگر تعداد مجاز داده‌های پرت صفر باشد، مسئله به همان مسئله‌ی kـ مرکز تبدیل می‌گردد. گونه‌ی مشابهی با مسئله‌ی kـ مرکز با داده‌های پرت تعریف می‌گردد که در آن تعدادی از نقاط نمی‌توانند به عنوان مرکز انتخاب شوند. به این‌گونه، مسئله‌ی kـ مرکز با داده‌های پرت و داده‌های ممنوعه(برای قرار گیری مرکز در آن‌ها) تعریف می‌شود. در مرجع [۴]، ثابت شده است که این مسئله در حالت کلی با ضریب کم‌تر از ۳ قابل تقریب‌پذیر نیست مگر آن‌که $P = NP$ باشد.

قضیه‌ی ۳ـ۴ *فرض کنید نقطه‌هایی دلخواه از یک متریک دلخواه داده شده‌اند. مسئله‌ی kـ 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 مسئله‌ی kـ مرکز در فضای اقلیدسی است [۹]. در واقع در این راه‌حل، مسئله‌ی بیشینه پوشش رأسی 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 مسئله‌ی kـ مرکز با داده‌های پرت و مراکز ممنوعه، دارای الگوریتمی با ضریب تقریب کم‌تر از ۳ داشته باشد (کوتاه‌ترین مسیر بین دو رأس غیر مجاور در دو بخش مختلف)، آن‌گاه می‌توان مسئله‌ی بیشینه پوشش رأسی را در زمان چندجمله‌ای حل نمود. برای مشاهده‌ی اثبات کامل‌تر می‌توانید به مرجع [۴] مراجعه کنید.* □

الگوریتمی که چریکار[۱۳] سایرین در این مقاله ارائه داده‌اند یک الگوریتم ۳ـ تقریب برای مسئله‌ی kـ مرکز با داده‌های پرت است. در ابتدا، الگوریتم چریکار، تمام شعاع‌های ممکن را پیدا می‌کند. در حالت مسئله‌ی kـ مرکز گسسته، کافی است تمام فاصله‌های بین دوبه‌دوی نقاط را به عنوان کاندیدا در نظر گرفت که تعدادشان از مرتبه‌ی $\mathcal{O}(n^{2})$ است و کافی است برروی گزینه‌های به دست آمده، جست‌وجوی دودویی زد. در صورتی که مسئله‌ی kـ مرکز در حالت پیوسته مد نظر باشد، به ازای هر $d+1$ نقطه‌ی دلخواه، یک کاندید اضافه می‌گردد که تعدادشان از مرتبه‌ی $\mathcal{O}(n^{(d+1)})$ است. حال اگر ما شعاع r داشته باشیم که r بزرگ‌تر مساوی شعاع بینه باشد، چریکار یک الگوریتم ساده با شعاع حداکثر $3r$ ارائه می‌دهد 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 شعار بهینه کم‌تر است.

تعریف ۳ـ۲ *به ازای هر نقطه‌ی $v_i \in V$، G_i(E_i 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 توپ گسترش‌یافته 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 kـ مرکز با z داده‌ی پرت

۱: به ازای i از ۱ تا k:

۲: به ازای تمام نقاط، توپ‌ها و توپ‌های گسترش‌یافته را محاسبه کن.

۳: G_j را توپی در نظر بگیر که بیش‌ترین وزن را دارد(بیش‌ترین تعداد نقاط پوشش‌داده نشده را می‌پوشاند).

۴: توپ E_j را به عنوان توپ iاُم در نظر بگیر.

۵: تمام نقاط داخل E_j را به مجموعه نقاط پوشش‌داده شده اضافه کن.

۶: اگر همه‌ی نقاط به جز حداکثر zتای آن‌ها پوشانده شده بودند:

۷: r اولیه بزرگ‌تر مساوی r بهینه است. برگردان E_jهای انتخاب شده

۸: در غیر این صورت:

۹: r اولیه کوچک‌تر از r بهینه است.

---

[۱۳] Charikar

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 [۴] کنید. چریکار در ادامه، با استفاده از برنامه‌ریزی خطی[۱۴] و گردسازی[۱۵] جواب، یک الگوریتم ۳ـ تقریب برای حالت گسسته و یک ۴ـ تقریب برای حالت پیوسته ارائه می‌دهد. به علت عدم استفاده از روش برنامه‌ریزی خطی، از بیان جزئیات این قسمت صرف نظر می‌کنیم. مک‌کاتن[۱۶] 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 و در نظر گرفتن نقاط آزاد به عنوان نقاطی هنوز مطمئن نیستیم نقطه‌ی پرت هستند و نقاطی که مطمئن هستیم باید پوشانده شوند. این الگوریتم حافظه‌ای از مرتبه‌ی $O(\frac{kz}{\epsilon})$ مصرف می‌کند. برای مشاهده جزئیات بیش‌تر به مرجع [۵] مراجعه کنید.

تا به اینجا مسئله‌ی kـ مرکز را در حالت کلی بررسی کردیم. مسئله‌ی ۱ـ مرکز با داده‌های پرت به صورت جداگانه به‌وسیله‌ی ضرابی‌زاده و سایرین مورد بررسی قرار گرفته است [۲۷]. در الگوریتم ارائه شده برای حالتی که $z = ۱$ است، ضرابی‌زاده یک الگوریتم با ضریب تقریب $۱/۴۸ \leqslant \frac{\sqrt{۲}+۱}{۲} \times ۱/۲۲$ با حافظه‌ی مصرفی $O(\frac{d}{\epsilon^۳}\log\frac{۱}{\epsilon})$ ارائه داده است. به ازای zهای کلی، الگوریتم دیگری با حافظه‌ی مصرفی $O(d^۳z + \frac{d}{\epsilon^۳}\log\frac{۱}{\epsilon})$ و ضریب تقریب $۱/۷۳ \leqslant \sqrt{۲} \times ۱/۲۲$ ارائه‌شده 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[۱۷] تعریف می‌شود که هر نقطه از جویبارداده به محض ورود به آن اضافه می‌شود. در صورتی که حافظه‌ی میان‌گیر، پر گردد، یکی از نقاط ار حافظه استخراج شده و به الگوریتم زیرین داده می‌شود. الگوریتم زیرین یک الگوریتم جویبارداده برای مسئله‌ی ۱ـ مرکز بدون داده‌های پرت است. همان‌طور که در زیربخش مسئله‌ی kـ مرکز در حالت جویبارداده بیان شد، بهترین الگوریتم موجود یک الگوریتم با حافظه‌ی مصرفی $O(d^۳z + \frac{d}{\epsilon^۳}\log\frac{۱}{\epsilon})$ و ضریب تقریب ۱/۲۲ است.

عامل ثانویه‌ای که در ضریب تقریب نهایی الگوریتم تاثیر 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 شدند و O مجموعه نقاطی که به اشتباه از حافظه‌ی میان‌گیر استخراج شدند

[۱۴] Linear Programming
[۱۵] Rounding
[۱۶] McCutchen
[۱۷] Buffer

باشد، اگر داشته باشیم که

$$Meb((P - O_x) \cup O) \leqslant \beta Meb(P - O_x)$$

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

در حالت کلی z، ایده‌ی اصلی برای استخراج یک نقطه از حافظه‌ی میان‌گیر، نقطه‌ی مرکزی[۱۸] 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 $\beta \leqslant \sqrt{2}$ خواهد بود. برا مشاهده‌ی اثبات و جزئیات بیش‌تر به مرجع [۲۷] مراجعه کنید.

در فصل آتی، در ابتدا به پیشرفت‌هایی که برای مسئله‌ی ۱ ـ مرکز در حالت جویبارداده با داده‌های پرت در این پایان‌نامه ارائه شده است، خواهیم پرداخت. سپس برای مسئله‌ی ۲ ـ مرکز در حالت جویبارداده با داده‌های پرت، اولین کار موجود را ارائه می‌دهیم که بهبود قابل توجهی نسبت به حالت کلی است.

[۱۸] Centerpoint

فصل ۴

نتایج جدید

در این فصل نتایج جدید به‌دست‌آمده در پایان‌نامه توضیح داده می‌شود. این فصل در سه بخش تهیه شده است. بخش اول به بیان مقدمات و نمادگذاری‌های مورد نیاز برای بخش‌های بعدی می‌پردازد. در بخش دوم، راه‌حل‌های ارائه‌شده برای مسئله‌ی ۱ ــ مرکز با z داده‌ی پرت در حالت جویبار داده مورد بررسی قرار می‌گیرد. این بخش به سه زیربخش تقسیم می‌شود.

در زیربخش اول، دو الگوریتم جدید ارائه می‌شود. الگوریتم اول، با مصرف حافظه‌ی $\mathcal{O}(z^{2}d)$، جوابی با ضریب تقریب ۲ ارائه می‌دهد که حافظه‌ی مصرفی الگوریتم ضرابی‌زاده و سایرین [۲۷] را در صورتی که ابعاد فضا بیش‌تر از z باشد بهبود می‌بخشد. از طرفی الگوریتم ارائه شده بسیار ساده‌تر از الگوریتم ضرابی‌زاده است. الگوریتم دوم یک الگوریتم با ضریب تقریب $1/8 + \epsilon$ و حافظه‌ی مصرفی $\mathcal{O}(\frac{z^{2}}{\epsilon})$ است.

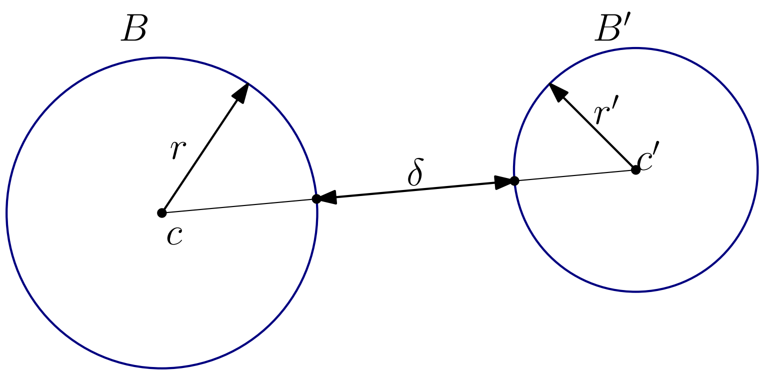
در زیر بخش دوم، به بررسی مسئله‌ی ۱ ــ مرکز پوشاننده بدون داده‌ی پرت می‌پردازیم، به‌طوری‌که نه تنها می‌خواهیم تمام نقاط ورودی پوشیده شود، بلکه می‌خواهیم کل توپ بهینه نیز به طور کامل پوشیده شود. برای این مسئله، یک الگوریتم با ضریب تقریب ۱/۷ ارائه می‌شود. در زیربخش سوم، با استفاده از زیربخش‌های قبلی، الگوریتمی با ضریب تقریب ۱/۷ برای مسئله‌ی ۱ ــ مرکز با z داده‌ی پرت ارائه می‌دهیم(برای حالتی که z ثابت است) .

در بخش سوم، مسئله‌ی ۲ ــ مرکز با zــ داده‌ی پرت مورد بررسی قرار می‌گیرد. ایده‌ی اصلی این بخش، تقسیم‌بندی مسئله به دو حالت است که حاصل، دو الگوریتم متفاوت می‌شود که به صورت

موازی اجرا می‌شوند. هر دوی این الگوریتم‌ها ضریب تقریب $\epsilon + ۱/۸$ دارند و حافظه‌ی مصرفی در کل برابر است با $O(dz^{۲}(d^{۲} + \frac{z}{\epsilon}))$. این اولین الگوریتم ارائه‌شده بعد از الگوریتمی که با ضریب تقریب $۴+\epsilon$ برای k کلی ارائه شده است، می‌باشد که بهبود قابل توجهی محسوب می‌شود.

۴ـ۱ نمادگذاری‌ها و تعاریف اولیه

در این قسمت، 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 در این فصل مورد بررسی قرار می‌گیرد.



شکل ۴ـ۱: 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 توپی ارجاع دهیم از نماد $r(B)$ و هر گاه خواستیم به مرکز یک توپ 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 ۴ـ۱ 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:*

$$\delta(B, B') = \max\{۰, \|cc'\| - r - r'\}$$

- *دو توپ دلخواه $B(c, r)$ و $B'(c', r')$ را αـ تفکیک‌شده گوییم اگر داشته باشیم:*

$$\delta(B, B') \geqslant \alpha. \max\{r(B), r(B')\}$$

- زمانی که می‌خواهیم در مورد توپ بهینه‌ی ۱ ــ مرکز صحبت کنیم از $B^*(c^*, r^*)$ یا $MEB(c^*, r^*)$ استفاده می‌کنیم. برای مسئله‌ی ۲ ــ مرکز نیز از $B_1^*(c_1^*, r^*)$ و $B_2^*(c_2^*, r^*)$ برای نشان دادن دو دایره‌ی بهینه مسئله‌ی ۲ ــ مرکز برای مجموعه‌ای از نقاط استفاده می‌کنیم.

- مجموعه نقاط P داده شده است. k ــ دورترین نقطه از $p \in P$ نقطه‌ای از P است که فاصله‌اش از نقطه‌ی p، k ــ امین بزرگ‌ترین فاصله را در بین تمام نقاط P داراست.

علاوه بر نمادگذاری‌های بالا، در بخش‌های بعدی، بعضی از تعاریف رایج در هندسه‌ی محاسباتی مورد استفاده قرار می‌گیرد. فرض کنید مجموعه‌ی n ــ عضوی P از نقاط در $\mathbb{R}^d$ داده شده است. نقطه‌ی $c \in \mathbb{R}^d$ را نقطه‌ی مرکزی[۱] مجموعه‌ی P می‌گویند، اگر هر نیم‌صفحه‌ی[۲] شامل c، حداقل شامل $\lceil \frac{n}{d+1} \rceil$ نقطه از نقاط P باشد. مرجع [۲۸] ثابت کرده است که هر مجموعه‌ی متناهی از نقاط در فضای d ــ بعدی، دارای یک نقطه‌ی مرکزی است. مشاهده‌ی زیر نتیجه‌ی مستقیم این گزاره است.

مشاهده‌ی ۴ ــ ۱ مجموعه‌ی P با $k(d+1)$ نقطه در فضای d ــ 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 ــ بعدی، $O(d)$ حافظه مصرف می‌کند و عملیات عادی بر روی نقاط نیز $O(d)$ زمان می‌برد.

۴ ــ ۲ مسئله‌ی ۱ ــ مرکز در حالت جویبار داده

در این بخش به بررسی گونه‌های مختلفی از مسئله‌ی ۱ ــ مرکز در حالت جویبار داده می‌پردازیم. مباحث این بخش، به صورت سه زیربخش دسته‌بندی شده است. در زیربخش اول مسئله‌ی ۱ ــ مرکز با داده‌های پرت، مورد بررسی قرار می‌گیرد. در زیرقسمت دوم مسئله‌ی ۱ ــ مرکز پوشاننده مورد بررسی قرار می‌گیرد و در نهایت در بخش سوم، با استفاده از نتایج دو بخش قبلی، مسئله‌ی ۱ ــ مرکز با تعداد ثابتی داده‌ی

[۱] Centerpoint
[۲] Half Space

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.

۴ــ۲ــ۱ مسئله‌ی ۱ــ مرکز با داده‌های پرت در حالت جویبار داده

در این زیربخش، دو الگوریتم کاملا متفاوت برای مسئله‌ی ۱ــ مرکز ارائه می‌شود که نسبت به الگوریتم‌های موجود ساده‌تر هستند و حافظه‌ی مصرفی کم‌تری دارند. از طرفی دیگر، دارای ویژگی‌هایی هستند که با استفاده از آن‌ها، در فصول بعدی، الگوریتم تقریبی برای مسئله‌ی ۲ــ مرکز ارائه می‌شود.

الگوریتم تقریبی با ضریب تقریب ۲

در این قسمت، یک الگوریتم ساده‌ی جویبارداده با ضریب تقریب ۲ برای مسئله‌ی ۱ــ مرکز با داده‌ی پرت ارائه می‌شود. در این الگوریتم، از ایده‌ی موازی‌سازی[۳] استفاده می‌شود که به وفور در بخش‌های آتی مورد استفاده قرار می‌گیرد. در الگوریتم ۴ شبه‌کد[۴] الگوریتم ارائه‌شده آمده است. الگوریتم، جویبارداده‌ی 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 (دقیقا z دورترین نقطه از نقطه‌ی اول را نمی‌پوشاند).

قضیه‌ی ۴ــ۲ *الگوریتم ۴ با فرض این‌که نقطه‌ی اول جویبار داده، داده‌ی پرت نیست یک الگوریتم تقریبی با ضریب تقریب ۲ برای مسئله‌ی ۱ــ 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:*

$$\|cp\| \leqslant \|cc^*\| + \|c^*p\| \leqslant ۲r^*$$

[۳] Parallelization

[۴] Pseudocode

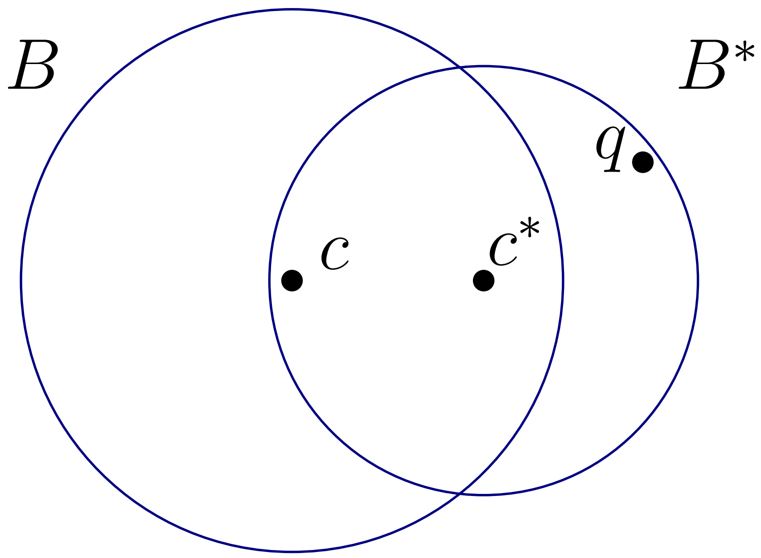
الگوریتم ۴ الگوریتم با ضریب تقریب ۲ برای مسئله‌ی ۱ ـ مرکز با z داده‌ی پرت

c را اولین نقطه از جویبارداده‌ی P قرار بده.
توپ $B(c, ۰)$ را در نظر بگیر.
حافظه‌ی میانگیر خالی Q را در نظر بگیر.
به ازای هر p در مجموعه‌ی P:
    اگر $p \notin B$:
        p را به Q اضافه کن.
        اگر $|Q| = z + ۱$:
            q را نزدیک‌ترین نقطه‌ی Q به مرکز c در نظر بگیر.
            q را از Q حذف کن.
            B را با توپ $B(c, \|cq\|)$ جایگزین کن.
B را برگردان

از طرفی، از بین $z + ۱$ 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 ۴ـ۲ نگاه کنید). با توجه به گزاره‌ی گفته‌شده، $\|cq\| \leqslant ۲r^*$ است. از طرفی چون شعاع جواب الگوریتم ۴ به اندازه‌ی فاصله‌ی c از $z + ۱$ـ دورترین نقطه از c است، بنابراین شعاع جواب الگوریتم نیز کم‌تر مساوی $۲r^*$ است و بنابراین الگوریتم ۴ یک الگوریتم ۲ـ تقریب برای مسئله‌ی ۱ ـ مرکز با z داده‌ی پرت است.

□

الگوریتم ۴ به طور ضمنی فرض کرده است که نقطه‌ی اول، در جواب بهینه نقطه‌ی پرت نیست. برای حذف چنین فرضی، $z + ۱$ نمونه از الگوریتم ۴ به طور موازی اجرا می‌گردد به‌طوری‌که در هر کدام، یکی از $z + ۱$ نقطه‌ی اول به آن به عنوان نقطه‌ی اول جویبار داده به الگوریتم داده می‌شود و بقیه نقاط در ادامه می‌آید. به وضوح، در بین $z + ۱$ نقطه‌ی اول، حتما یک نقطه وجود دارد که در جواب بهینه داده‌ی پرت نیست. بنابراین جواب آن نمونه از الگوریتم، یک ۲ـ تقریب برای جوابه بهینه است و در نتیجه، کوچک‌ترین توپ بین $z + ۱$ نمونه‌ی موازی، همواره یک ۲ـ تقریب برای جواب بهینه است. با توجه به این‌که پیچیدگی حافظه‌ی الگوریتم ۴ برای یک نمونه از مرتبه‌ی $O(zd)$ است و زمان به‌روزرسانی آن از



شکل ۴ـ۲: اثبات قضیه‌ی ۴ـ۲

مرتبه‌ی $\mathcal{O}(d+\log z)$ است، نتیجه زیر به دست می‌آید.

قضیه‌ی ۴ـ۳ *برای یک جویبارداده از نقاط در فضای dـ بعدی، الگوریتم ۴ یک ۲ـ تقریب برای مسئله‌ی ۱ـ مرکز با z داده‌ی پرت با حافظه‌ی مصرفی $\mathcal{O}(z^۲d)$ و زمان به‌روزرسانی $\mathcal{O}(zd+z\log z)$ ارائه می‌دهد.*

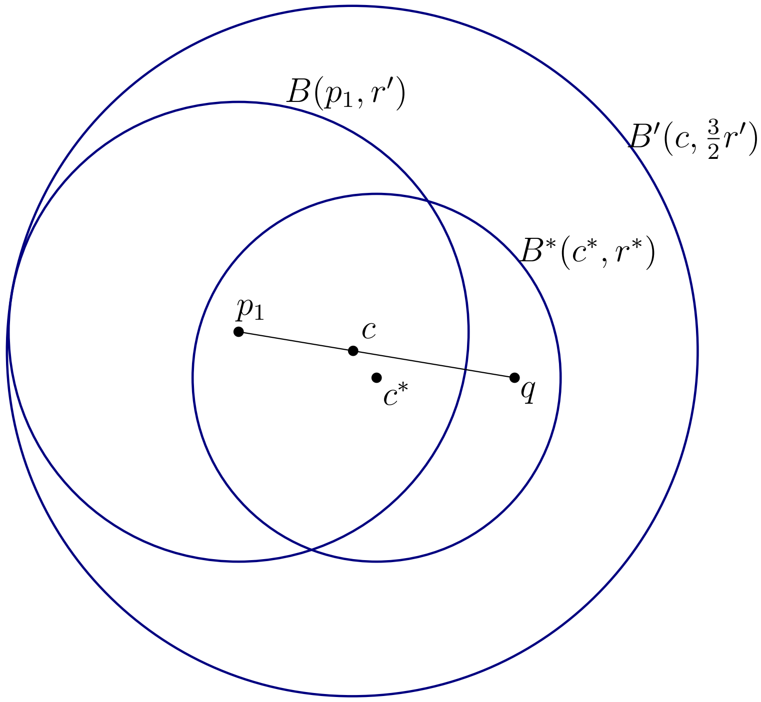
الگوریتم تقریبی با ضریب تقریب ۱/۸

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

$$۱/۲r^* \leqslant r' \leqslant (۱/۲+\frac{۲\epsilon}{۳})r^*$$

با فرض داده شدن r'، الگوریتم ۵، یک توپ با شعاع حداکثر $\frac{۳}{۲}r'$ ارائه می‌دهد که حداکثر z نقطه از جویبارداده 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 $z+۱$ نمونه از الگوریتم ارائه شده را به طور موازی اجرا نموده و از بین $z+۱$ 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:



شکل ۴ــ۳: گسترش توپ $B(c, r')$ در راستای نقطه‌ی q

لم ۴ــ۴ *همان‌طور که در شکل ۴ــ۳ نشان داده شده است، نقطه‌ی q از جویبارداده‌ی P را در نظر بگیرید به‌طوری‌که در توپ بهینه‌ی B^* قرار گرفته (داده‌ی پرت نیست) و فاصله‌ی آن از p_1 بزرگ‌تر مساوی r' باشد. نقطه‌ی c را در فاصله‌ی $\frac{1}{2}r'$ از p_1 برروی پاره‌خط p_1q در نظر بگیرید. ثابت می‌شود توپ $B'(c, \frac{3r'}{2})$، توپ B^* و $B(p_1, r')$ را به طور کامل می‌پوشاند (توپ B^* و $B(p_1, r')$ به طور کامل داخل B' قرار می‌گیرند). به چنین عملی گسترش توپ $B(p_1, r')$ در راستای نقطه‌ی q گفته می‌شود.*

اثبات. *طبق لم ۳ــ۳، مرکز توپ B^* که با c^* نشان داده می‌شود، حداکثر $0/8r^*$ از q فاصله دارد. برای هر نقطه‌ی s که در توپ $B(c, r')$ قرار می‌گیرد، داریم:*

$$\|sc\| \leqslant \|sp_1\| + \|p_1c\| \leqslant r' + \frac{1}{2}r' \leqslant \frac{3r'}{2}$$

از طرفی برای هر نقطه‌ی s داخل B^* داریم:

$$\|sc\| \leqslant \|sc^*\| + \|c^*c\| \leqslant r^* + 0/8r^* \leqslant \frac{3r'}{2}$$

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

□

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.

الگوریتم ۵ الگوریتم با ضریب تقریب ۱/۸ برای مسئله‌ی ۱ـ مرکز با z داده‌ی پرت

- فرض کنید r' تقریب برای $1/2r^*$ و z_0 تعداد نقاط پرت قبل ار گسترش B داده شده‌اند.
- توپ $B(p_1, r')$ را در نظر بگیر.
- حافظه‌ی میان‌گیر خالی Q را در نظر بگیر.
- به ازای هر p در مجموعه‌ی P:
  - اگر $p \notin B$:
    - p را به Q اضافه کن.
    - اگر B هنوز گسترش پیدا نکرده و $|Q| = z_0 + 1$:
      - توپ B را در راستای p گسترش بده.
      - به ازای هر $q \in Q$:
        - اگر $q \in B$:
          - q را از Q حذف کن.
    - اگر $Q = z + 1$:
      - از برنامه خارج شو.
- B را برگردان

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 کاندید 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 $1/2r^* \leqslant r' \leqslant (1/2 + \frac{2\epsilon}{3})r^*$ است، بنابراین یک جواب با ضریب تقریب $1/2 + \epsilon$ از جواب بهینه به دست آورده‌ایم. اگر حافظه‌ی میان‌گیر

پر شود، حتما یکی از اعضای آن وجود دارد که جزء داده‌های پرت نبوده (طبق اصل لانه‌کبوتری[۵]). 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 و مطمئن هستیم کل جواب بهینه را نیز می‌پوشاند.

پس از گسترش هر کدام از گزینه‌ها، کافی است در هر لحظه نگه‌داریم 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 تاست ($z+۱$ نقطه‌ی حافظه‌ی میان‌گیر به غیر از نقطه‌ای که در آن راستا توپ را گسترش داده‌ایم)، ممکن است خارج توپ گسترش‌یافته قرار بگیرند و نیازی به نگه‌داشتن نقاط قبلی نیست. اگر در ادامه‌ی جویبارداده تعداد نقاط خارج از توپ گسترش‌یافته بیش از z عدد گردد، یا 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 حالتی وجود دارد که فرض برای آن درست است) و هر کدام یک جواب با شعاع حداکثر $\frac{۳}{۲}r'$ ارائه می‌دهند که اگر از بین آن‌ها توپ با شعار کمینه را به عنوان جواب نهایی بدهیم، مطمئنا یک جواب با تقریب حداکثر $۱/۸+\epsilon$ از جواب بهینه ارائه داده‌ایم.

تا به این‌جا الگوریتمی ارائه دادیم که با فرض داشتن r' و داده‌ی پرت نبودن $p_۱$، با مصرف حافظه‌ی $O(z)$ و زمان به‌روزرسانی $O(z)$ در هر لحظه می‌تواند یک $۱/۸+\epsilon$ ـ تقریب از جواب بهینه بدهد.

تنها قسمتی که مورد بررسی قرار نگرفته است، نحوه‌ی به‌دست‌آوردن 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' به‌دست‌آمده و الگوریتم ۵، یک $۱/۸+\epsilon$ 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 ـ تقریب (به طور مثال الگوریتم ۲ ـ تقریب ارائه شده در قسمت قبلی)، یک r به‌دست آوریم. طبق الگوریتم استفاده شده داریم:

$$r^* \leqslant r \leqslant \alpha r^*$$

[۵] Pigeonhole Principle

حال اگر بازه‌ی $[۰, ۱/۲r]$ را به

$$m = \left\lceil \frac{۳}{۲} \times ۱/۲ \times \alpha \times \frac{۱}{\epsilon} \right\rceil$$

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

$$\frac{۱/۲r}{m} = \frac{۲}{۳} \times \frac{r}{\alpha} \times \epsilon \leqslant \frac{۲\epsilon}{۳} r^*$$

از طرفی چون $۱/۲r \leqslant ۱/۲r^*$ است، بنابراین یکی از این بازه‌ها $۱/۲r^*$ 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 αــ تقریب برای مسئله‌ی ۱ ــ مرکز با z داده‌ی پرت، به ازای سرهای تمام بازه‌ها، الگوریتم ۵ را اجرا کنیم و از بین گزینه‌هایی که باقی می‌مانند کوچک‌ترین توپ را به عنوان جواب نهایی بدهیم. با توجه به این‌که برای سر یکی از بازه‌ها، r در شرایط $۱/۲r^* \leqslant r \leqslant (۱/۲ + \frac{۲\epsilon}{۳})r^*$ صدق می‌کند، در نتیجه یکی از گزینه‌ها یک $۱/۸ + \epsilon$ ــ تقریب برای جواب بهینه است و در نتیجه توپ با شعاع کمینه نیز همین ضریب تقریب را تضمین می‌کند.

تنها قسمتی که نیاز به دقیق شدن دارد، قسمت تک‌گذره کردن الگوریتم است. همان‌طور که گفته شد، از الگوریتم ۴ که الگوریتمی ۲ ــ تقریب است، برای پیدا کردن r' استفاده می‌کنیم. فرض کنید r_i برابر شعاع الگوریتم ۲ ــ تقریب برای جویبارداده تا iاُمین عنصر جویبار داده باشد. به وضوح دنباله‌ی r_i یک دنباله صعودی است. به ازای هر i، k را عدد صحیحی در نظر بگیرید که رابطه‌ی زیر برقرار باشد:

$$۲^{k-۱} \leqslant r_i \leqslant ۲^k$$

با توجه به k بالا $l_i = ۲^k$ قرار دهید. به وضوح طبق رابطه‌ی گفته شده، $l_i \leqslant ۲r_i$ است و در نتیجه l_i یک ۴ ــ تقریب برای مسئله‌ی ۱ ــ مرکز با z داده‌ی پرت است.

حال کافی است بازه‌ی $[۰, ۱/۲l_i]$ را به $m = \left\lceil \frac{۷/۲}{\epsilon} \right\rceil$ قسمت تقسیم کنیم. با این تقسیم‌بندی، طول هر بازه، $t_i = \frac{۱/۲l_i}{m}$ می‌شود و مجموعه سر بازه‌ها برابر

$$R_i = \{j \times t_i \mid ۱ \leqslant j \leqslant m\}$$

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

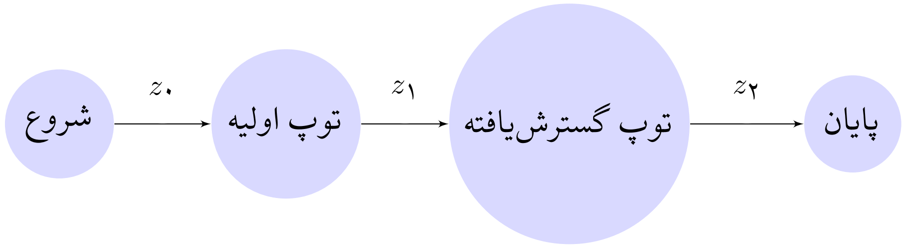
حال کافی است که در هر لحظه m نمونه از الگوریتم ۵ را به ازای هر $r \in R_i$ به صورت موازی اجرا نماییم. به ازای اضافه شدن نقطه‌ی p_i، اگر $l_i = l_{i-۱}$ باشد، $R_i = R_{i-۱}$ است و در نتیجه بدون هیچ

تغییری کافی است p_i را به تمام نمونه‌های موازی اضافه کنیم. در حالتی که $l_i > l_{i-1}$ باشد، مجموعه‌ی R_i را می‌توان به دو زیرمجموعه تقسیم نمود. اعضایی از $r \in R_i$ 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 اعضایی، کافی است، نمونه معادل آن را در R_{i-1} بدون هیچ تغییری پیدا کرد و نقطه‌ی p_i را به آن اضافه کرد. اگر $r \in R_i$ باشد و در R_{i-1} نباشد، در نتیجه $r \geqslant 1/2 l_{i-1} \geqslant l_{i-1}$ 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 برروی نقاط جویبارداده تاکنون اجرا نماییم، کافی است الگوریتم را به ازای نقاط داخل حافظه‌ی میان‌گیر اجرا نموده و مطمئن هستیم که بقیه‌ی نقاط به علت قرار گیری داخل $B(p_1, r)$، تاثیری در روند اجرای الگوریتم نخواهند داشت.

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 $\mathcal{O}(zd)$ زمان می‌برد. در هر مرحله هم حداکثر m نمونه‌ی جدید ساخته می‌شود، بنابراین زمان به‌روزرسانی نمونه‌ها در هر مرحله حداکثر $\mathcal{O}(\frac{zd}{\epsilon})$ است. از طرفی در هر لحظه mz (z به خاطر عدم اطمینان از نقطه‌ی دومی که داخل B^* قرار می‌گیرد و m به علت عدم اطمینان از محل قرارگیری r' در بازه‌ها) نمونه موازی از الگوریتم ۵ در حال اجراست. بنابراین، زمان به‌روزرسانی نهایی الگوریتم برابر $\mathcal{O}(\frac{z^2 d}{\epsilon})$ است و حافظه‌ی مصرفی نیز متناسب با mz نمونه‌ی موازی از الگوریتم ۵ برابر $\mathcal{O}(\frac{zd}{\epsilon})$ است. با دخیل کردن امکان پرت نبودن نقطه‌ی اول جویبار داده، به قضیه‌ی زیر می‌رسیم:

قضیه‌ی ۴ـ۵ *الگوریتم ۵ با مصرف حافظه‌ی $\mathcal{O}(\frac{z^2 d}{\epsilon})$ و زمان به‌روزرسانی $\mathcal{O}(\frac{z^3 d}{\epsilon})$، در هر لحظه با صرف زمان اجرای $\mathcal{O}(\frac{z^2}{\epsilon})$ جوابی با ضریب تقریب $1/8 + \epsilon$ ارائه می‌دهد.*

اگر بخواهیم الگوریتم گفته شده را جمع‌بندی کنیم، الگوریتم همان‌طور که در شکل ۴ـ۴ نشان‌داده شده است. در ابتدا z. نقطه‌ی اول را به عنوان داده‌ی پرت در نظر می‌گیرد و سپس $z + 1$اُمین نقطه‌ی جویبار داده را به عنوان نقطه‌ی اول و مرکز توپ B به شعاع r' در نظر می‌گیرد. سپس z_1 نقطه‌ی اولی از ادامه‌ی جویبارداده که بیرون این توپ قرار می‌گیرند را به عنوان داده‌ی پرت در نظر گرفته و به ازای



شکل ۴ـ۴: نحوه‌ی اجرای الگوریتم ۵

$z_1 + 1$اُمین نقطه‌ی خارج B، آن را در همان راستا گسترش می‌دهد. سپس z_2 نقطه‌ی دیگر از ادامه‌ی جویبارداده که خارج $B-$گسترش‌یافته قرار می‌گیرند را نیزبه عنوان داده‌ی پرت در نظر می‌گیرد، حال اگر نقطه‌ی دیگری در جویبارداده وجود داشته باشد که خارج B بیفتد با توجه به اینکه $z = \sum_{i=0}^{2} z_i$ است، تعداد نقاط پرت از z بیش‌تر شده و نشان می‌دهد یکی از فرض‌های اولیه اشتباه بوده و در نتیجه، گزینه حذف می‌گردد.

۴ـ۲ـ۲ مسئله‌ی ۱ـ مرکز پوشاننده در حالت جویبار داده

در این زیر قسمت به بررسی مسئله‌ی تقریبا جدیدی می‌پردازیم. در ابتدا به تعریف دقیق مسئله می‌پردازیم:

تعریف ۴ـ۱ *مجموعه نقاط P داده شده‌اند. به توپ B یک αـ تقریب برای مسئله‌ی ۱ـ مرکز پوشاننده گویند اگر نه تنها تمام نقاط P را بپوشاند، بلکه توپ بهینه‌ی ۱ـ مرکز این نقاط که با B^* نشان داده می‌شود را نیز به طور کامل می‌پوشاند و شعاع آن، حداکثر α برابر شعاع توپ بهینه باشد.*

اگر این مسئله را در حالت جویبار داده در نظر بگیریم، هدف نگه‌داری مجموعه‌ای هسته که بتوان با استفاده از آن، در هر لحظه یک αـ تقریب از مسئله‌ی ۱ـ مرکز در حالت پوشاننده ارائه داد. برای این مسئله دو الگوریتم ارائه می‌دهیم. در الگوریتم اول، الگوریتم ۱/۸ـ تقریب ارائه شده برای مسئله‌ی ۱ـ مرکز با z داده‌ی پرت را به گونه‌ای تغییر می‌دهیم که الگوریتمی با ضریب تقریب ۱/۸ برای مسئله‌ی ۱ـ مرکز پوشاننده در حالت جویبار داده ارائه دهد. در الگوریتم دوم، با استفاده از الگوریتمی دلخواه برای مسئله‌ی ۱ـ مرکز در حالت جویبارداده به عنوان جعبه‌ی سیاه[۶]، یک الگوریتم برای مسئله‌ی ۱ـ مرکز پوشاننده در حالت جویبارداده ارائه می‌دهد.

[۶] Black Box

الگوریتم $\epsilon + ۱/۸$ ــ تقریب برای مسئله‌ی ۱ ــ مرکز پوشاننده در حالت جویبارداده

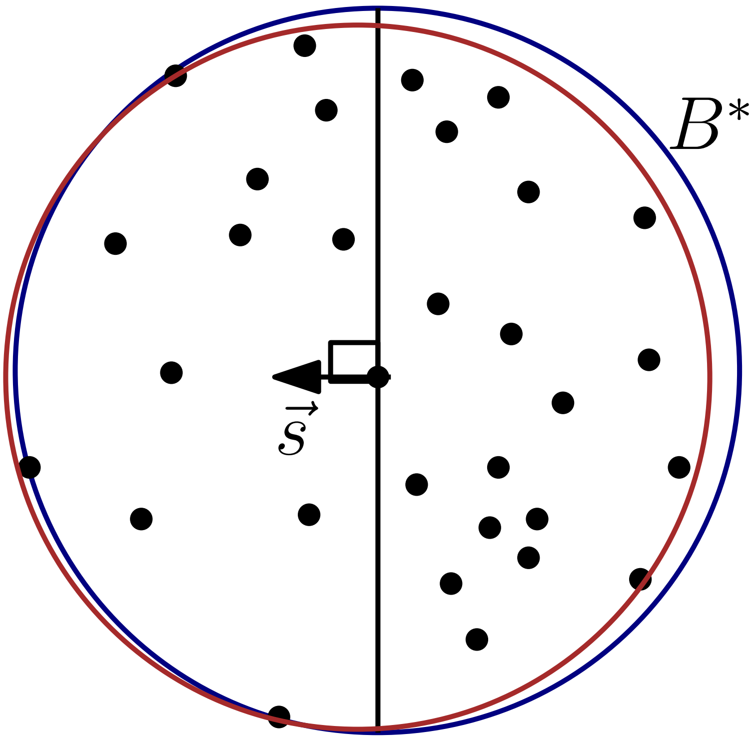
اگر الگوریتم ۵، را با $z = ۰$ برروی جویبار داده اجرا کنیم، با مصرف حافظه و زمان به‌روزرسانی از مرتبه‌ی $\mathcal{O}(\frac{d}{\epsilon})$ می‌تواند یک جواب ۱/۸ ــ تقریب از جواب بهینه برای مسئله‌ی ۱ ــ مرکز ارائه می‌دهد. توجه کنید که برای حالتی که $z = ۰$ است، تنها لازم است m نمونه از الگوریتم اجرا نمود که هر نمونه نیز $\mathcal{O}(d)$ 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 نشانش می‌دهیم)، داریم:

$$r_m \leqslant (۱/۲ + \frac{۲\epsilon}{۳})r^*$$

زیرا مطمئن هستیم r' ای که در بازه‌ی $[۱/۲r^*, (۱/۲ + \frac{۲\epsilon}{۳})r^*]$ است در بین گزینه‌ها قرار دارد. حال اگر توپی که با استفاده از r'_m ساخته شده باشد، شعاعی برابر با $\frac{۳}{۲}r'_m$ داشته باشد، مطمئن هستیم که B^* را به طور کامل می‌پوشاند و از طرفی شعاعش حداکثر ۱/۸ ــ برابر r^* است. اما اگر توپ گسترش نیافته باشند، یک توپ داریم که تمام نقاط را می‌پوشاند و شعاعش حداکثر ۱/۲ ــ برابر شعاع بهینه است. برای ادامه، نیاز به به دو لم زیر داریم:

لم ۴ــ۶ *فرض کنید مجموعه نقاط P از نقاط در فضای $\mathbb{R}^d$ داده شده‌اند. B^* را توپی 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 ۴ــ۵ نشان داده شده است، فرض کنید نیم‌کره‌ای از B^* وجود داشته باشد که در پوسته‌ی آن هیچ نقطه‌ای از P قرار ندارد. بردار عمود بر صفحه‌ای که کره‌ی B^* را به دو نیم کره تقسیم می‌کند را $\vec{s}$ در نظر بگیرید (در جهت 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 برروی نیم‌کره قرار نمی‌گیرد و پوسته‌ی نیم‌کره‌ی مقابل نیز کاملا خالی می‌شود. بنابراین می‌توان شعاع توپ را کاهش داد و به توپ قهوه‌ای رنگ که شعاع کم‌تری نسبت به B^* دارد رسید که تمام نقاط را می‌پوشاند، که با بهینه بودن B^* تناقض دارد. بنابراین فرض



شکل ۴ـ۵: اثبات لم ۴ـ۶

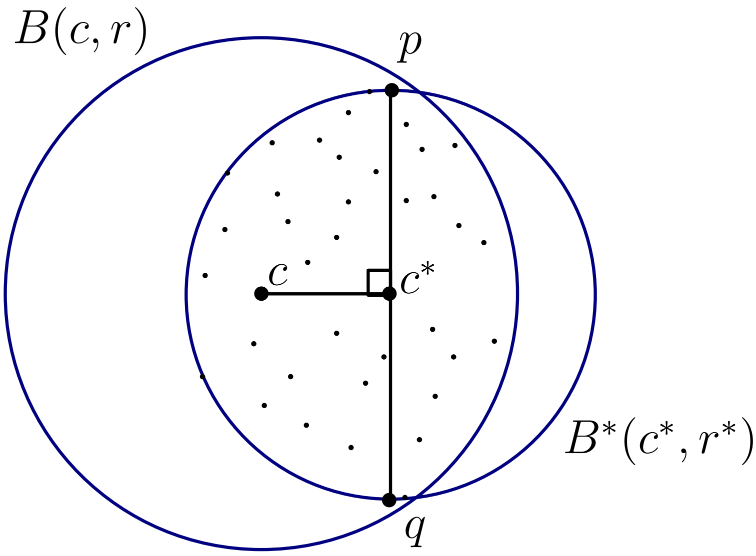
اولیه مبنی بر وجود نیم‌کره‌ای با پوسته‌ی خالی اشتباه بوده است.

□

لم ۴ـ۷ *فرض کنید مجموعه نقاط P از نقاط در فضای $\mathbb{R}^d$ داده شده‌اند. $B^*(c^*, r^*)$ را توپی با شعاع کمینه در نظر بگیرید که تمام نقاط را می‌پوشاند. توپ $B(c, r)$ با شعاعی αr^* در نظر بگیرید که تمام نقاط P را می‌پوشاند. ثابت می‌شود اگر شعاع توپ B را $\sqrt{۲}$ برابر کنیم، کل توپ B^* را نیز می‌پوشاند.*

اثبات. همان‌طور که در شکل ۴ـ۶ نشان داده شده است، هر دوی B و B^* کلیه‌ی نقاط را می‌پوشانند. حال صفحه‌ی عمود بر پاره‌خط واصل $\vec{c^*c}$ 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 $\vec{c^*c}$ بر صفحه‌ی رسم شده)، بنابراین یا به طور کامل داخل B قرار می‌گیرد یا بیرون آن. 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 گرفته و از طرفی طبق لم ۴ـ۶، حداقل یک نقطه‌ی s از P برروی پوسته‌ی این نیم‌کره قرار دارد، که

می‌توان نتیجه گرفت s به وسیله‌ی B پوشانده نمی‌شود. تناقض، پس دایره‌ی D به طور کامل داخل B قرار می‌گیرد.



شکل ۴ـ۶: اثبات لم ۴ـ۷

نقطه‌ی دلخواه q برروی این دایره را در نظر بگیرید. چون این نقطه برروی پوسته‌ی B^* قرار دارد، بنابراین داریم:

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

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

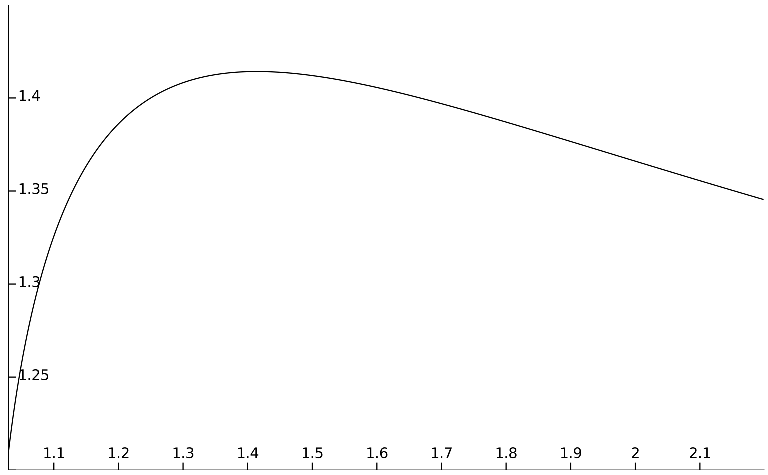
$$\|cq\| \leqslant r = \alpha r^*$$

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

$$\|cc^*\| = \sqrt{\|cq\|^{۲} - \|c^*q\|^{۲}} \leqslant \sqrt{\alpha^{۲} - ۱}r^*$$

حال با توجه به این‌که $\alpha \geqslant ۱$ است، اگر شعاع دایره‌ی B را $\sqrt{\alpha^{۲}-۱}r^*$ افزایش دهیم، دایره‌ی B^* را به طور کامل می‌پوشاند. در واقع برای افزایش شعاع به این میزان کافی است، شعاع را $\frac{۱+\sqrt{\alpha^{۲}-۱}}{\alpha}$ برابر کنیم. اگر مطابق شکل ۴ـ۷ نمودار این تابع را رسم کنیم، می‌بینیم که حداکثر تابع در نقطه‌ی $\sqrt{۲}$ و برابر $\sqrt{۲}$ خواهد بود. توجه کنید که اگر $\alpha \leqslant \sqrt{۲}$ باشد، تابع در بازه‌ی $[۱,\alpha]$ مقداری کم‌تر از $\sqrt{۲}$ به خود می‌گیرد و همان‌طور که در نمودار پیداست، چون تابع در این بازه صعودی است، مقدار بیشینه در خود α به دست می‌آید.

□



شکل ۴ـ۷: نمودار تابع $\frac{۱+\sqrt{\alpha^{۲}-۱}}{\alpha}$

با توجه به لم ۴ـ۷، برای حالتی که توپ گسترش پیدا نکرده است، اگر شعاعش را $\sqrt{۲}$ برابر کنیم، تضمین می‌کند که دایره‌ی بهینه را پوشانده است. با توجه به این‌که در این حالت شعار توپ کم‌تر مساوی $۱/۲r^*$ است، با $\sqrt{۲}$ برابر کردن شعاعش به توپی با شعاعی حداکثر ۱/۷ برابر r^* دست خواهیم یافت. بنابراین در هر دو حالت، توپی را بر می‌گردانیم که توپ بهینه را به طور کامل می‌پوشاند، شعاع آن حداکثر $۱/۸+\epsilon$ برابر جواب بهینه است.

الگوریتم ۱/۷ـ تقریب برای مسئله‌ی ۱ـ مرکز پوشاننده در حالت جویبارداده

در این قسمت با استفاده از لم ۴ـ۷، الگوریتمی با ضریب تقریب ۱/۷ برای مسئله‌ی ۱ـ مرکز پوشاننده در حالت جویبارداده ارائه می‌دهیم. ایده‌ی اصلی این الگوریتم، استفاده از یک الگوریتم αـ تقریب برای مسئله‌ی ۱ـ مرکز در حالت جویبار داده است و افزایش شعاع آن در زمان پاسخ‌گویی به پرسمان برای پوشش کامل توپ بهینه. همان‌طور که در فصل کارهای پیشین ذکر شده است، بهترین الگوریتم موجود برای مسئله‌ی ۱ـ مرکز در حالت جویبارداده، الگوریتم ارائه شده به وسیله‌ی آگاروال با حافظه‌ی مصرفی و زمان به‌روزرسانی $\mathcal{O}(d)$ و ضریب تقریب ۱/۲۲ است. حال اگر جواب این الگوریتم را بخواهیم افزایش بدهیم، طبق لم ۴ـ۷، باید شعاع آن را $\frac{۱+\sqrt{۱/۲۲^{۲}-۱}}{۱/۲۲}$ برابر کنیم، که توپی با شعاع حداکثر ۱/۷ برابر شعاع توپ بهینه ارائه می دهد. زمان اجرا و حافظه‌ی مصرفی این الگوریتم همانند الگوریتم آگاروال، $\mathcal{O}(d)$ است.

۴ـ۲ـ۳ مسئله‌ی ۱ـ مرکز با داده‌های پرت در حالت جویبار داده با تعداد داده‌های پرت ثابت

در این قسمت، با استفاده از الگوریتم‌هایی که در قسمت قبل برای مسئله‌ی ۱ـ مرکز پوشاننده در حالت جویبارداده ارائه شده است، الگوریتم جدیدی برای مسئله‌ی ۱ـ 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 ۴ـ ۸ *مجموعه‌ی P از نقاط در فضای $\mathbb{R}^d$ در نظر بگیرید. حداکثر $(d+۱)^z$ حالت برای انتخاب نقاط پرت این مجموعه وجود دارد. در واقع حداکثر $(d+۱)^z$ زیرمجموعه‌ی z عضوی از P وجود دارد که دایره‌ی با شعاع کمینه‌ای که سایر نقاط P را می‌پوشاند، هیچ کدام از اعضای زیر مجموعه‌ی انتخابی را نپوشاند.*

اثبات. *از استقراء برای اثبات لم استفاده می‌کنیم.*

- ***پایه:*** *حکم برای $z=۰$ 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 $z=k-۱$ حداکثر $(d+۱)^z$ حالت برای انتخاب زیرمجموعه داده‌های پرت وجود دارد.*
- ***حکم:*** *ثابت می‌کنیم به ازای هر مجموعه‌ی دلخواه P در $\mathbb{R}^d$ و $z=k$ حداکثر $(d+۱)^z$ 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 برروی پوسته‌ی B^* قرار دارند را در نظر بگیرید. از بین اعضای S می‌توان زیرمجموعه‌ی حداکثر $d+۱$ عضوی S' را انتخاب کرد به طوری که دایره‌ی محاطی که کم‌ترین شعاع آن همان B^* گردد (در فضای dـ بعدی، هر توپ را با حداکثر $d+۱$ نقطه روی پوسته‌ی آن می‌توان مشخص کرد).*

  زیرمجموعه‌ای دلخواه O برای 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. بنابراین 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 استقرا $(d+1)^{z-1}$ حالت دارد. در نتیجه در کل تعداد حالات انتخاب O حداکثر برابر $(d+1)^z$ است.

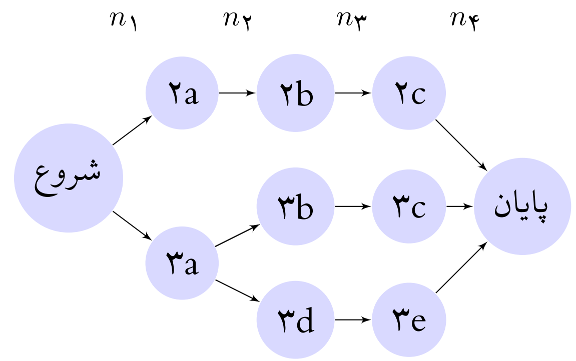
□

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 می‌کنیم و دومین حالت، حالتی است که آن را به مجموعه نقاط غیر پرت حالت اولیه اضافه می‌کنیم. توجه کنید که مجموعه نقاط پرت، یک حافظه با اندازه‌ی حداکثر z و مجموعه نقاط غیر پرت، همان اجرای الگوریتم ۱ ـ 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 ـ مرکز با z داده‌ی پرت در حالت جویبار داده را با ضریب تقریب ۱/۷ (حاصل استفاده از الگوریتم ارائه شده در قسمت قبل برای مسئله‌ی ۱ ـ مرکز پوشاننده در حالت جویبارداده) حل می‌نماید. حافظه‌ی مصرفی و زمان به‌روزرسانی الگوریتم ارائه شده از مرتبه‌ی $\mathcal{O}(d \times (d+1)^z)$ است.

۴ـ۳ مسئله‌ی ۲ ـ مرکز با داده‌های پرت در حالت جویبار داده

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



شکل ۴ـ۸: حالت‌های الگوریتم کیم و آهن [۱]

همان‌طور که در بخش نمادگذاری‌ها ذکر شده بود، $B_1^*(c_1^*, r^*)$ و $B_2^*(c_2^*, r^*)$ را توپ‌های جواب بهینه برای مسئله‌ی ۲ ـ مرکز با z داده‌ی پرت برای جویبارداده‌ی 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 زیربخش اول به بررسی حالت $\delta^* \leqslant \alpha r^*$ و در زیربخش 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.

۴ـ۳ـ۱ حالت $\delta^* \leqslant \alpha r^*$

ایده‌ی اصلی این بخش، تغییر الگوریتم ارائه شده به وسیله‌ی کیم[۷] و آهن[۸] [۱] که در اصل برای نگه‌داری مجموعه‌ی هسته برای مسئله‌ی ۲ ـ مرکز در حالت جویبارداده با ضریب تقریب $\epsilon + ۱/۸$ ارائه شده است، حاصل می‌گردد. الگوریتم آهن و کیم، در واقع مبنای الگوریتم‌های ارائه شده در همین پایان‌نامه برای مسئله‌ی ۱ ـ مرکز با داده‌های پرت در حالت جویبارداده با ضریب تقریب $\epsilon + ۱/۸$ و مسئله‌ی ۱ ـ مرکز پوشاننده در حالت جویبارداده با ضریب تقریب $\epsilon + ۱/۸$ است. تغییرات اعمال شده نیز بسیار شبیه عملکرد الگوریتم $\epsilon + ۱/۸$ ـ تقریب برای مسئله‌ی ۱ ـ مرکز با داده‌ی پرت است. بنابراین، در این قسمت، برای جلوگیری از تکرار، در ابتدا گام‌های اصلی الگوریتم کیم را بیان کرده و سپس تغییراتی که در الگوریتم جدید مورد نیاز است را ذکر می‌کنیم.

همان‌طور که در شکل ۴ ـ ۸ نشان داده شده است، الگوریتم کیم و آهن، دارای ۱۰ حالت مختلف

[۷] Kim
[۸] Ahn

است. متناسب با نقاطی که تا کنون در جویبارداده آمده‌اند، الگوریتم در یکی از حالت‌های بالا قرار دارد. 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 جویبارداده وارد شود که در هیچ کدام از توپ‌های کاندید قرار نگیرد.

الگوریتم کیم و آهن، از گره‌ی شروع، شروع می‌کند و با رسیدن نقاط جدید از جویبارداده در طول گراف مطابق با یال‌ها جابه‌جا می‌گردد. در بعضی از حالت‌ها، بیش‌تر 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 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[۹] 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 ــ 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' کاملا مشابه r' استفاده شده در الگوریتم $1/8+\epsilon$ ــ تقریب برای مسئله‌ی ۱ ــ مرکز با داده‌های پرت در همین پایان‌نامه است).

الگوریتم اصلی

در این بخش تغییراتی که برروی الگوریتم کیم و آهن ارائه دادیم را بیان می‌کنیم. الگوریتم ارائه شده، کاملا مشابه الگوریتم ارائه شده برای مسئله‌ی ۱ ــ 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 کاندید 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 کاندید 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، کاندیدها به گونه‌ای

[۹] Deterministic

تغییر می‌کنند که کاندیدهای قبلی را به طور کامل می‌پوشانند.

بنابراین تنها وجود نقاطی در جویبارداده مهم هستند که خارج توپ‌های کاندید قرار می‌گیرند. با توجه به این‌که این نقاط تنها باعث افزایش شعاع توپ‌های کاندید 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 نقات پرت را در نظر بگیریم. چون گراف تغییر حالات[۱۰]، 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 در نظر گرفتن تمام حالات ممکن، تمام ۴ـ تایی‌های صحیح نامنفی ($n_۱, \cdots, n_۴$) که $\sum_{i=۱}^{۴} n_i = z$ را در نظر بگیریم. به راحتی می‌توان نشان داد که تعداد چنین ۴ـ تایی‌هایی از مرتبه‌ی $\mathcal{O}(z^۳)$ است.

شبه‌کد الگوریتم ارائه شده در الگوریتم ۶ نشان داده شده است. به ازای تمام حالت‌های ممکن برای $n_۱$ تا $n_۴$ و هر مسیر مجاز از بین سه مسیر موجود بین گره‌ی شروع تا پایان، الگوریتم یک جواب کاندید ($B_۱, B_۲$) 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 کاندید $B_۱ = B(p_۱, r')$ و $B_۲ = \emptyset$ شروع می‌کند که معادل حالت شروع الگوریتم کیم و آهن است. پس از ورود هر نقطه‌ی p از جویبارداده، در ابتدا بررسی می‌شود که نقطه‌ی مورد نظر در توپ‌های کاندید 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 jاُم 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 کاندید ($B_۱, B_۲$) را مطابق با الگوریتم کیم و اهن، به‌روزرسانی می‌کنیم. توجه کنید که برای انتقال حالت مطابق الگوریتم کیم و آهن، علاوه بر نقطه‌ی p، ما مسیر حرکت را نیز می‌دهیم که به طور قطعی، حالت بعدی مشخص شود.

زمانی که تمام نقاط P پردازش شدند، اگر هنوز به گره‌ی پایان وارد نشده‌ایم، توپ‌های کاندید را به عنوان یک جواب به مجموعه جواب اضافه می‌کنیم. در غیر این صورت مطابق عملکرد الگوریتم کیم و

[۱۰] Transition Graph

الگوریتم ۶ مسئله‌ی ۲ ـ مرکز در حالت نزدیک

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

مجموعه جواب S را برابر $\emptyset$ قرار بده.

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

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

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

$counter_i$ را برابر صفر قرار بده.

B_1 را $B(p_1, r')$ قرار بده.

B_2 را مجموعه‌ی $\emptyset$ قرار بده.

متغیر j را برابر ۱ قرار بده. ◁ *متغیر j عمق حالت را مشخص می‌کند.*

به ازای هر $p \in P$:

اگر $p \notin B_1 \cup B_2$:

$counter_j$ را یک عدد افزایش بده.

اگر $counter_j > n_j$:

j را یک عدد افزایش بده. ◁ *در مسیر π به حالت بعدی برو*

توپ‌های کاندید (B_1, B_2) را با توپ‌های کاندید حاصل از انتقال حالت مطابق مسیر π در الگوریتم کیم و آهن جایگزین کن.

اگر $j \leqslant 4$:

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

کم‌ترین شعاع داخل مجموعه‌ی S را برگردان

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 [۱] ثابت کرده‌اند، جوابی که با این روش محاسبه می‌شود دارای شعاعی حداکثر $\frac{3}{2}r^*$ است، با فرض اینکه $\delta^* \leqslant \alpha r^*$ 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 داشتن $1/2r^* \leqslant r' \leqslant (1/2 + \frac{2\epsilon}{3})r^*$، قضیه زیر برقرار است:

قضیه‌ی ۴ـ۹ *به ازای $1/2r^* \leqslant r' \leqslant (1/2 + \frac{2\epsilon}{3})r^*$ داده شده و با فرض $\delta^* \leqslant \alpha r^*$، الگوریتم ۶ یک جواب $1/8 + \epsilon$ ـ تقریب برای جواب بهینه‌ی مسئله‌ی ۲ ـ مرکز با z داده‌ی پرت ارائه می‌دهد. حافظه‌ی مصرفی این الگوریتم از مرتبه‌ی $\mathcal{O}(dz^3)$ و زمان به‌روزرسانی/پاسخ‌گویی به پرسمان آن، از مرتبه‌ی $\mathcal{O}(dz^3)$ است (با فرض پرت نبودن داده‌ی اول).*

در زیر بخش بعدی، نحوه‌ی پیدا کردن r' مناسب را مورد بررسی قرار می‌دهیم.

پیدا کردن r'

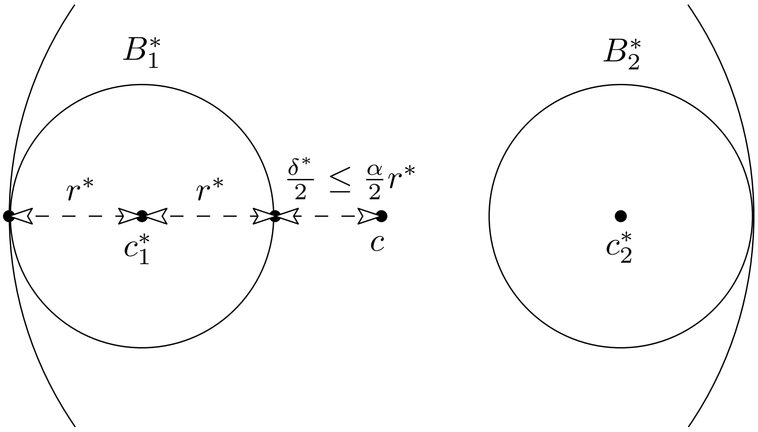
در این زیر بخش، نشان می‌دهیم که چگونه r' مناسبی را پیدا کنیم که در رابطه‌ی

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

صدق کند. لم زیر ایده‌ی اصلی را بیان می‌کند.

لم ۴ـ۱۰ *مجموعه نقاط P در فضای $\mathbb{R}^d$ داده شده است. یک جواب بهینه برای مسئله‌ی ۱ ـ مرکز با z داده‌ی پرت برای مجموعه نقاط P، با فرض $\delta^* \leqslant \alpha r^*$، یک $(2 + \frac{\alpha}{2})$ـ تقریب برای مسئله‌ی ۲ ـ مرکز با z داده‌ی پرت برای مجموعه‌ی نقاط P ارائه می‌دهد.*

اثبات. *فرض کنید r_1^* و r^* به ترتیب شعاع بهینه برای مسئله‌ی ۱ ـ مرکز و ۲ ـ مرکز با z داده‌ی پرت برای مجموعه نقاط P باشد. به وضوح $r_1^* \leqslant r^*$ است، زیرا هر جواب درست B^* برای مسئله‌ی ۱ ـ مرکز با z داده‌ی پرت، یک جواب درست (B^*, B^*) برای مسئله‌ی ۲ ـ مرکز با z داده‌ی پرت ارائه می‌دهد. حال $B_1^*(c_1^*, r^*)$ و $B_2^*(c_2^*, r^*)$ را دو توپ جواب مسئله‌ی ۲ ـ مرکز با z داده‌ی پرت در نظر بگیرید. c را نقطه‌ی میانی پاره‌خط واسط مراکز c_1^* و c_2^* در نظر بگیرید (همان‌طور که در شکل ۴ـ۹ نشان داده شده*



شکل ۴ـ۹: اثبات لم ۴ـ۱۰

است). به وضوح، $B(c, \frac{\delta}{۲} + ۲r^*)$ هر دوی $B_۱^*$ و $B_۲^*$ را می‌پوشاند. بنابراین یک جواب قابل قبول برای مسئله‌ی ۱ ـ مرکز با z داده‌ی پرت است، و در نتیجه داریم:

$$r_۱^* \leqslant (۲ + \frac{\alpha}{۲})r^*$$

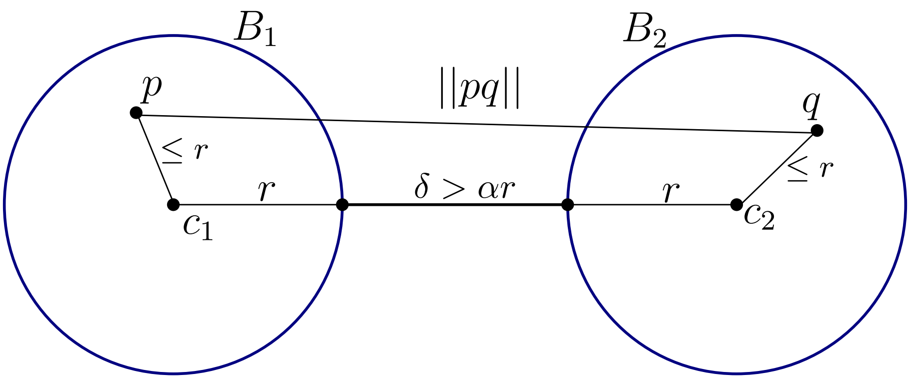
□

حال کافی است، به طور کاملا مشابه با الگوریتم ۵ از الگوریتم ۴ برای تقریب r^* استفاده کنیم و با تقسیم بندی بازه‌ی $[۰, ۱/۲r]$ به $m = \left\lceil \frac{۱/۸}{\epsilon} \times (۴ + \alpha) \right\rceil$ زیربازه، 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:

قضیه‌ی ۴ـ۱۱ *اگر $\delta^* \leqslant \alpha r^*$ باشد، یک $(۱/۸ + \epsilon)$ ـ تقریب برای مسئله‌ی ۲ ـ مرکز با z داده‌ی پرت با مصرف حافظه‌ی $O(\frac{dz^۴}{\epsilon})$ و زمان به‌روزرسانی $O(\frac{dz^۵}{\epsilon})$ قابل ارائه است.*

۴ـ۳ـ۲ حالت $\delta^* > \alpha r^*$

در این بخش، الگوریتمی با ضریب تقریب ۱/۸ برای حالتی که دو توپ بهینه بیش از αr^* یک‌دیگر فاصله دارند. با دو مشاهده‌ی ساده شروع می‌کنیم.



شکل ۴ـ۱۰: اثبات مشاهده‌ی ۴ـ۱۲

مشاهده‌ی ۴ـ۱۲ *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:*

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

اثبات. *همان‌طور که در شکل ۴ـ۱۰ می‌بینید، با استفاده از نامساوی مثلثی رابطه‌ی زیر برقرار است:*

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

از طرفی با توجه با نحوه‌ی تعریف δ، می‌دانیم فاصله‌ی هر زوج دلخواه از (B_1, B_2) از جمله p و q، حداقل δ است. در نتیجه داریم:

$$\delta \leqslant \|pq\| \leqslant 4r + \delta$$

که با تقسیم طرفین بر δ به حکم مسئله می‌رسیم.

□

مشاهده‌ی ۴ـ۱۳ *فرض کنید B_1 و B_2 دو توپ با فاصله‌ی δ باشند و B یک توپ با شعاع کم‌تر از $\frac{\delta}{2}$ 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 ۴ـ۱۴ *فرض کنید B_1^* و B_2^* دو توپ αـ جداپذیر باشند $(\alpha > 4)$. اگر p نقطه‌ی دلخواه از B_1^* و S زیرمجموعه‌ی $z+1$ عضوی از P شامل دورترین نقاط از p باشد، آن‌گاه $S \cap B_2^*$ ناتهی است.*

اثبات. *از برهان خلف برای اثبات استفاده می‌کنیم. با برهان خلف فرض می‌کنیم که خلاف حکم مسئله برقرار و $S \cap B_2^*$ تهی باشد. چون $|S| = z + 1 > z$ است، بنابراین حداقل عضوی از S وجود دارد که جزء داده‌های پرت در جواب بهینه نیست و چون S اشتراکش با B_2^* تهی است، بنابراین آن عضو داخل B_1^* قرار دارد و مجموعه‌ی $S \cap B_1^*$ ناتهی 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 $s \in P \setminus S$، $\|ps\| \leqslant \|pq\|$ 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:*

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

بنابراین با توجه به مشاهده‌ی ۴ـ۱۳، $B \cap B_2^*$ تهی است. بنابراین $P \cap B_2^*$ تهی است و در نتیجه B_2^* تهی است، که با بهینه بودن B_1^* و B_2^* تناقض دارد.

□

لم ۴ـ۱۵ *فرض کنید p نقطه‌ای دلخواه در B_1^* باشد و q، $z + 1$ـ دورترین نقطه از p باشد. آن‌گاه $\delta^* > \frac{\alpha}{\alpha+4}\|pq\|$.*

اثبات. *با استفاده از لم ۴ـ۱۴، نقطه‌ی $q' \in B_2^*$ وجود دارد به‌طوری‌که $\|pq\| \leqslant \|pq'\|$. بنابراین با توجه به مشاهده‌ی ۴ـ۱۲ داریم:*

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

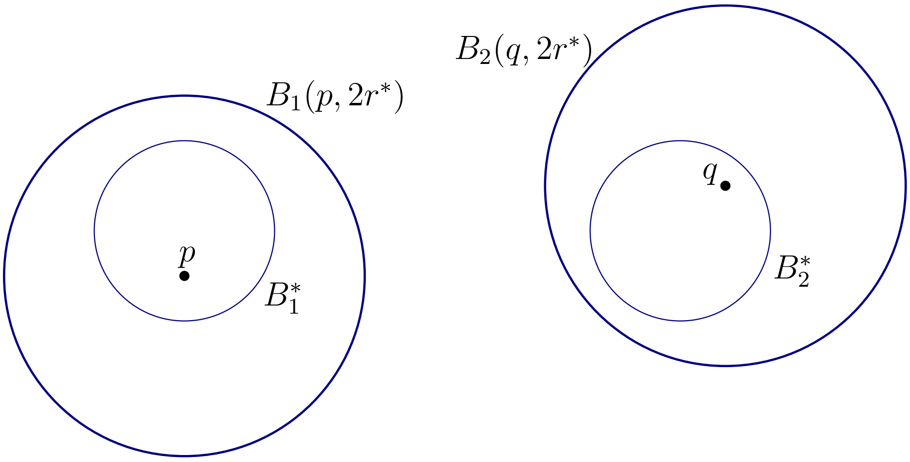
□

لم ۴ـ۱۶ *دو نقطه‌ی $p \in B_1^*(c_1, r^*)$ و $q \in B_2^*(c_2, r^*)$ در نظر بگیرید، آن‌گاه $B_1^* \subset B(p, 2r^*)$ و $B_2^* \subset B(q, 2r^*)$ 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:*

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

و در نتیجه $B_1^* \subset B(p, 2r^*)$ است (به شکل ۴ـ۱۱ مراجعه کنید). به طور مشابه ثابت می‌شود $B_2^* \subset B(q, 2r^*)$ است. با توجه به این‌که حداکثر z نقطه‌ی پرت خارج B_1^* و B_2^* قرار می‌گیرد، در نتیجه اثبات کامل است.



شکل ۴ـ۱۱: اثبات لم ۴ـ۱۶

□

لم ۴ـ۱۷ *فرض کنید S زیرمجموعه‌ای از P با اندازه‌ی حداقل $(d+1)(z+1)$ باشد که توسط توپی با شعاع کم‌تر از $\frac{\delta^*}{2}$ پوشانده می‌شود. آن‌گاه c_p نقطه‌ی مرکزی نقاط S، یا داخل B_1^* قرار می‌گیرد یا داخل B_2^* 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 ۴ـ۱۳، B دقیقا 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 ۴ـ۱ حداقل $z+1$ نقطه‌ی دیگر خارج B_1^* قرار می‌گیرند، که 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 قسمت، الگوریتم اصلی برای حالتی که $\delta^* > \alpha r^*$ است. در هر لحظه، الگوریتم نقاط P از جویبارداده 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_2^* را به طور کامل بپوشاند و Buffer تنها شامل تعدادی از

نقاط پرت در جواب بهینه است. توجه کنید که $B_۱$ و $B_۲$ علاوه بر پوشش توپ متناظر در جواب بهینه، ممکن است تعدادی از نقاط پرت را نیز شامل شوند.

با شروع الگوریتم، مرکز توپ $B_۱$ را که با $c_۱$ نشان می‌دهیم برابر $p_۱$ قرار می‌دهد و $c_۲$ را از میان نقاطی که تا کنون پردازش‌شده است به عنوان کاندیدی برای مرکز $B_۲$ انتخاب می‌کند. الگوریتم همچنین دو متغیر δ و r را در طول جویبارداده 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 $۲r^*$ است.

شبه کد الگوریتم، در الگوریتم ۷ آورده شده است. به محض ورود نقطه‌ی p از جویبارداده‌ی P، الگوریتم ارائه شده، سعی می‌کند آن را به توپ $B_۱$ یا $B_۲$ اضافه کند. عمل اضافه کردن نقاط به توپ‌های $B_۱$ و $B_۲$ به ترتیب به وسیله‌ی توابع $AddToB_1$ و $AddToB_2$ انجام می‌شود. اگر نقطه‌ی p، در هیچ کدام از توپ‌ها قرار نگیرد، به $Buffer$ اضافه می‌شود. تابع $AddToB_1$ نقطه‌ی p را به $B_۱$ اضافه می‌کند اگر فاصله‌ی p از $c_۱$ کم‌تر مساوی δ باشد. تابع $AddToB_2$ نقطه‌ی p را به مجموعه‌ی $B_۲$ اضافه می‌کند اگر نقطه‌ی p از $c_۲$ حداکثر r فاصله داشته باشند. هر دوی توابع، مقدارهای δ و r را در صورت لزوم تغییر می‌دهند که ناورداهای داخل لم ۴ـ۱۸ برقرار بماند.

زمانی که $Buffer$ سرریز می‌کند (در الگوریتم ۷)، الگوریتم یکی از دو عملیات زیر را وابسته به این‌که اندازه‌ی $B_۲$ چقدر است، انجام می‌دهد. اگر اندازه‌ی $|B_۲|$ بزرگ‌تر مساوی $(z+۱)(d+۱)$ باشد، تمام اعضای $B_۲$ را به $B_۱$ اضافه می‌شود و $B_۲$ خالی می‌گردد. در غیر این صورت، $c_۲$ در صورتی که قبلاً تعیین شده باشد، به $B_۱$ اضافه می‌شود و نقطه‌ی دیگری از $B_۲ \cup Buffer \setminus \{c_۲\}$ به عنوان $c_۲$ جدید انتخاب می‌گردد. حلقه‌ی مذکور، حداکثر $\mathcal{O}(dz)$ بار اجرا می‌شود، زیرا پس از اولین اجرا، مطمئن هستیم که T حداکثر $z+(z+۱)(d+۱)$ نقطه دارد و در هر مرحله حداقل یک عنصر از آن حذف می‌گردد ($c_۲$). توجه داشته باشید که هر نقطه حداکثر یک‌بار به عنوان $c_۲$ انتخاب می‌شود، بنابراین حلقه‌ی مذکور، به طور سرشکن[۱۱] یک‌بار به ازای هر نقطه از جویبارداده اجرا می‌شود.

برای تحلیل، علاوه بر $c_۲$، نیاز به نقطه‌ی مرکزی به نام c_p داریم. زمانی که $|B_۲| < (z+۱)(d+۱)$، آن‌گاه c_p همان $c_۲$ است و در غیر این صورت، c_p را نقطه‌ی مرکزی $(z+۱)(d+۱)$ نقطه‌ی اولی که به

[۱۱] Amortized

الگوریتم ۷ الگوریتم برای ۲ـ مرکز در حالت دور

c_1 را برابر p_1 قرار بده.
r و δ را برابر صفر قرار بده.
به ازای هر نقطه‌ی $p \in P$:
اگر p_1 قابل اضافه شدن به B_1 و B_2 نبود: ▷ *از دو تابع $AddToB_1$ و $AddToB_2$ به‌ترتیب برای اضافه کردن نقطه به B_1 و B_2 استفاده کنید.*
p را به Buffer اضافه کن.
تا وقتی $z \geqslant |\text{Buffer}|$:
اگر $|B_2| \geqslant (d+1)(z+1)$:
B_1 را برابر $B_1 \cup B_2$ قرار بده.
B_2 را برابر مجموعه‌ی تهی قرار بده.
در غیر این صورت:
اگر c_2 مشخص شده است:
c_2 را به B_1 اضافه کن.
متغیر محلی T را برابر $\text{Buffer} \cup B_2 \setminus \{c_2\}$ قرار بده.
B_2 را تهی قرار بده.
c_2 را $(z+1)$ـ امین دورترین نقطه از c_1 در T قرار بده.
r را برابر با $\frac{2}{\alpha}\|c_1c_2\|$ قرار بده.
به ازای هر $p \in T$:
p را به B_2 اضافه کن.
Buffer را برابر $T \setminus B_2$ قرار بده.

الگوریتم ۸ تابع اضافه‌کننده نقطه به B_1

اگر حداقل $z+1$ نقطه پردازش شده است:
  q را $(z+1)$ـ دورترین نقطه از c_1 در نقاطی که تا کنون آمده‌اند در نظر بگیر.
در غیر این صورت:
  q را برابر c_1 در نظر بگیرد.
δ را برابر $\frac{\alpha}{\alpha+4}\|c_1q\|$ قرار بده.
اگر $p \in B(c_1,\delta)$:
  p را به مجموعه‌ی B_1 اضافه کن.
  برگردان true
برگردان false

B_2 اضافه می‌شوند قرار می‌دهیم.

لم ۴ـ۱۸ *ثابت‌های حلقه‌ی زیر در طول اجرای الگوریتم حفظ می‌شوند:*

۱. $\delta < \delta^*$

۲. $r \leqslant \frac{\delta}{2}$

۳. $B_1 \cap B_2^* = \emptyset$

۴. *اگر $c_p \in B_2*$ باشد، آن‌گاه:*

(آ) $2r^* \leqslant r$

(ب) $B_2 \cap B_2^* = \emptyset$

(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 این‌که $z+1$ نقطه از جویبار داده پردازش می‌شود، تابع $AddToB_1$ مقدار δ را به $\frac{\alpha}{\alpha+4}\|c_1q\|$ افزایش می‌دهد، که*

الگوریتم ۹ تابع اضافه‌کننده نقطه به B_2

اگر c_2 تعیین شده باشد و $p \in B(c_2, r)$:
  p را به B_2 اضافه کن.
  اگر $|B_2| = (d+1)(z+1)$:
    r را $(2 + \frac{2}{\alpha})$ برابر کن.
    به ازای $p \in B(c_2, r)$:
      اگر $p \in B(c_2, r)$:
        p را به B_2 اضافه کن.
        p را از Buffer حذف کن.
در غیر این صورت:
  برگردان true
برگردان false

در آن q، $(z+1)$ـ دورترین نقطه از c_2 در جویبارداده است. از طرفی چون $c_1 \in B_1^*$ است، طبق لم ۴ـ۱۵ داریم $\delta < \delta^*$

۲. *زمانی که متغیر c_2 در الگوریتم ۷ تعیین می‌شود، $(z+1)$ـ دورترین نقطه از c_1 در بین اعضای $T \subset P$ است و r برابر $\frac{2}{\alpha}\|c_1c_2\|$ است. اگر q، $(z+1)$ـ دورترین نقطه از c_1 در جویبارداده در آن لحظه باشد، آن‌گاه $\|c_1q\| \leqslant \|c_1c_2\|$ است. با فرض این‌که $\alpha \geqslant 16$ باشد، داریم:*

$$\frac{2}{\alpha}\|c_1c_2\| \leqslant \frac{1}{6} \times \frac{\alpha\|c_1q\|}{\alpha+4} \leqslant \frac{\delta}{6}$$

و در نتیجه:

$$r \leqslant (2+\frac{2}{\alpha}) \times \frac{2}{\alpha}\|c_1c_2\| \leqslant 3 \times \frac{2}{\alpha}\|c_1c_2\| \leqslant \frac{\delta}{2}$$

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 ۴ـ۱۹ *اگر c_2 تعیین شده باشد، آن‌گاه $B(c_p, \frac{2}{\alpha}\|c_1c_p\|) \subset B_2(c_2, r)$ است.*

اثبات. اگر $|B_۲| < (d+۱)(z+۱)$ باشد، با توجه به این‌که $c_p = c_۲$ است و $r = \frac{۲}{\alpha}\|c_۱c_۲\|$، در نتیجه $B_۲ = B(c_p, \frac{۲}{\alpha}\|c_۱c_p\|)$ است و حکم برقرار است. در حالتی که اندازه‌ی $B_۲$ به $(d+۱)(z+۱)$ می‌رسد، نقطه‌ی c_p به نقطه‌ی مرکزی نقاط $B_۲$ انتقال می‌یابد و r، $(۲+\frac{۲}{\alpha})$ برابر می‌گردد. از طرفی چون نقطه‌ی مرکزی نقاط داخل $B_۲$، داخل $B_۲$ قرار می‌گیرد، بنابراین $c_p \in B(c_۲, \frac{۲}{\alpha}\|c_۱, c_۲\|)$ است. در نتیجه زمانی داریم:

$$\|c_۲c_p\| \leqslant \frac{۲}{\alpha}\|c_۱c_۲\|$$

و در نتیجه داریم:

$$\frac{۲}{\alpha}\|c_۱c_p\| \leqslant \frac{۲(\|c_۱c_۲\| + \|c_۲c_p\|)}{\alpha} \leqslant \frac{۲}{\alpha}\|c_۱c_۲\|(۱+\frac{۲}{\alpha})$$

که نتیجه می‌دهد:

$$B(c_p, \frac{۲}{\alpha}\|c_۱c_p\|) \subset B_۲(c_۲, \frac{۲}{\alpha}\|c_۱c_۲\|(۲+\frac{۲}{\alpha}))$$

□

حال با استفاده از ادعای ۴ـ۱۹، حکم را ثابت می‌کنیم. نقطه‌ی p از جویبار داده را در نظر بگیرید که به $B_۱$ اضافه شده است. این نقطه در دو شرایط می‌تواند به $B_۱$ اضافه شده باشد. حالت اول، زمانی است که تابع $AddToB_1$ صدا زده می‌شود. در این تابع، نقطه‌ی p تنها زمانی به $B_۱$ اضافه می‌شود که در فاصله‌ی δ از $c_۱$ قرار داشته باشد. با توجه به ناوردای ۱، مطمئن هستیم که $\delta < \delta^*$ و در نتیجه $\|c_۱p\| < \delta^*$ است و در نتیجه $p \notin B_۲^*$. در حالت دوم، در الگوریتم ۷، زمانی که حافظه‌ی میان‌گیر *Buffer* سر ریز می‌کند و $B_۲$ خالی نیست، الگوریتم وابسته به اندازه‌ی $B_۲$ عمل می‌کند. اگر اندازه‌ی $B_۲$ هنوز به $(d+۱)(z+۱)$ نرسیده باشد، آن‌گاه $c_p = c_۲$ است. در این حالت، الگوریتم ۴ـ۱۹، $c_۲$ را به $B_۱$ اضافه می‌کند. با فرض خلف فرض کنید که $c_p \in B_۲^*$ باشد. آن‌گاه با توجه ناوردای ۲ و ۴ قسمت (آ) داریم:

$$۲r^* \leqslant r \leqslant \frac{\delta}{۲} \leqslant \frac{\delta^*}{۲}$$

در نتیجه با توجه به لم ۴ـ۱۶، حداکثر باید z نقطه خارج $B_۱ \cup B_۲$ قرار بگیرد، که با سرریز شدن حافظه‌ی میان‌گیر *Buffer* تناقض دارد. در حالتی که $|B_۲| \geqslant (d+۱)(z+۱)$ است، آن‌گاه c_p، نقطه‌ی مرکزی $(d+۱)(z+۱)$ اولین نقاطی است که به $B_۲$ اضافه شده‌اند. در این حالت،

تمام نقاط $B_۲$ به $B_۱$ اضافه می‌شود. با استفاده از ناوردای ۲ ، داریم:

$$r \leqslant \frac{\delta}{۲} < \frac{delta^*}{۲}$$

در نتیجه، طبق لم ۴ــ۱۷ ، $c_p \in B_۱^*$ یا $c_p \in B_۲^*$ است. با فرض خلف، فرض کنید که $c_p \in B_۲^*$ است. در این حالت، با توجه به ناوردای ۴ قسمت (آ) ، و ادعای ۴ــ۱۹ ، $B_۲$ توپ $B(c_p, \frac{۲}{\alpha}\|c_۱c_p\|)$ را می‌پوشاند و $r \leqslant ۲r^*$ است. بنابراین کاملا مشابه حالتی که $|B_۲| < (d+۱)(z+۱)$ ، با سرریز شدن حافظه‌ی میان‌گیر Buffer تناقض دارد و در نتیجه $c_p \in B_۱^*$ و در نتیجه $B_۲ \cap B_۲^* = \emptyset$ است و اضافه کردن آن به $B_۱$ صورت ناوردا را نقض نمی‌کند.

۴. (آ) اگر $c_۱ \in B_۱^*$ باشد و $c_p \in B_۲^*$ باشد، طبق مشاهده‌ی ۴ــ۱۳ ، اگر $c_۱ \in B_۱^*$ باشد و $c_p \in B_۲^*$ باشد، آن‌گاه

$$۱ \leqslant \frac{\|c_۱c_p\|}{\delta^*} \leqslant \frac{\|c_۱c_p\|}{\alpha r^*}$$

و در نتیجه:

$$۲r^* \leqslant \frac{۲}{\alpha}\|c_۱c_p\|$$

اگر $|B_۲| < (d+۱)(z+۱)$ باشد، $c_p = c_۲$ است و با توجه به الگوریتم ۷ ، $r = \frac{۲}{\alpha}\|c_۱c_۲\|$ است و در نتیجه $r \leqslant ۲r^*$ است. اگر $|B_۲| < (d+۱)(z+۱)$ باشد، مشابه با ناردای ۲ داریم:

$$۲r^* \leqslant \frac{۲}{\alpha}\|c_۱c_p\| \leqslant (۱+\frac{۲}{\alpha})\frac{۲}{\alpha}\|c_۱c_۲\| \leqslant (۲+\frac{۲}{\alpha})\frac{۲}{\alpha}\|c_۱c_۲\| = r$$

(ب) بر اساس ناوردای ۴ قسمت (آ) ، اگر $c_p \in B_۲^*$ باشد، در نتیجه داریم:

$$۲r^* \leqslant r \leqslant \frac{\delta}{۲}$$

و $c_p \in B_۲$ است. حال با توجه به ناوردای ۱ و مشاهده‌ی ۴ــ۱۳ ، $B_۲$ تنها $B_۲^*$ را قطع می‌کند و در نتیجه $B_۲ \cap B_۱^* = \emptyset$.

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

$$۲r^* \leqslant r \leqslant \frac{\delta}{۲} < \frac{\delta^*}{۲}$$

و در نتیجه با توجه به لم ۴ــ۱۶ ، تمام نقاط داخل حافظه‌ی میان‌گیر Buffer یا خارج از $B_۱ \cup B_۲$ داده‌ی پرت هستند.

□

پاسخ‌گویی به پرسمان‌ها

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

- *اگر بدانیم دو توپ بهینه‌ی جواب مسئله‌ی ۲ـ مرکز با z داده‌ی پرت، αـ جداپذیر باشند، دو توپ هم‌شعاع پیدا کنید که همه‌ی نقاط به غیر از حداکثر z نقطه از نقاطی که تاکنون در جویبار داده آمده‌اند را بپوشاند.*

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 ناورداهای ۳ و ۴، اگر $c_p \in B_2^*$ 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 $c_p \in B_2^*$ اشتباه باشد و در نتیجه B_1، B_1^* و B_2، B_2^* را نپوشاند. برای برطرفی 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 ۲ـ مرکز با حداکثر z داده‌ی پرت ارائه می‌دهد.

فرض کنید C مجموعه‌ی نقاط کاندید برای c_2 باشد. با استفاده از ناوردای ۳، داریم $B_1 \cap B_2^* = \emptyset$. بنابراین، $(B_2 \cup Buffer) \cap B_2^* \neq \emptyset$ و در نتیجه وجود دارد c_2 در این مجموعه که می‌توان از روی آن، جواب با تقریب مناسبی برای ۲ـ مرکز با حداکثر z داده‌ی پرت به‌دست آورد. در لم ۴ـ۲۰، برای حالتی $|B_2| \geqslant (d+1)(z+1)$ است، نشان داده‌شده است که $(\{c_2\} \cup Buffer) \cap B_2^* \neq \emptyset$. بنابراین اگر $|B_2| \geqslant (d+1)(z+1)$ باشد، کافی است مجموعه‌ی $\{c_2\} \cup Buffer$ را به عنوان مجموعه‌ی نقاط کاندید برای c_2 در نظر گرفت. بنابراین اندازه‌ی مجموعه‌ی نقاط کاندید از $O(zd)$ است.

لم ۴ـ۲۰ *در هر لحظه، اگر $|B_2| \geqslant (d+1)(z+1)$ باشد و $c_p \notin B_2^*$ باشد، آن‌گاه $B_2 \cap B_2^* = \emptyset$.*

اثبات. *با استفاده از ناورداهای ۱ و ۲، داریم:*

الگوریتم ۱۰ پاسخ‌گویی به پرسمان

مجموعه‌ی solutions را برابر $\{\mathrm{MinCover}(B_1, B_2, \mathrm{Buffer})\}$ قرار بده.
اگر $|B_2| < (d+1)(z+1)$ است:
    مجموعه‌ی candidates را برابر Buffer قرار بده.
    مجموعه‌ی B_1 را برابر $B_1 \cup B_2$ قرار بده.
    مجموعه‌ی B_2 را تهی کن.
در غیر این صورت:
    مجموعه‌ی candidates را برابر $B_2 \cup \mathrm{Buffer} \setminus \{c_2\}$ قرار بده.
به ازای هر $c \in \mathrm{candidates}$:
    r را برابر $\frac{2}{\alpha}\|c_1 c\|$ قرار بده.
    B'_1 را برابر B_1 قرار بده.
    δ را برابر $\max\{\delta_0, r\}$ قرار بده.
    B'_2 و حافظه‌ی میانگیر Buffer' را برابر مجموعه‌ی تهی قرار بده.
    به ازای هر $p \in \mathrm{candidates}$:
        اگر اگر نقطه‌ی p به B'_1 و B'_2 اضافه نشد:
            p را به حافظه میانگیر Buffer' اضافه کن.
    $\mathrm{MinCover}(B'_1, B'_2, \mathrm{Buffer}')$ را به مجموعه‌ی solutions اضافه کن.
کم‌ترین عضو مجموعه‌ی solutions را برگردان.

الگوریتم ۱۱ محاسبه‌ی پوشش بهینه

ورودی: B_1 به عنوان توپی که با B_2^* اشتراک ندارد، B_2 به عنوان توپی که با B_1^* 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 $[(z - |\mathrm{Buffer}|) \cdots ۰]$:
۳: r_1 را برابر شعاع $\mathrm{Center}(B_1, k) - ۱$ قرار بده.
۴: r_2 را برابر شعاع $\mathrm{Center}(B_2, z - |\mathrm{Buffer}| - k) - ۱$ قرار بده.
۵: بیشینه‌ی r_1 و r_2 را به مجموعه‌ی solutions اضافه کن.
۶: کمینه عضو solutions را به عنوان خروجی برگردان.

$$r \leqslant \frac{\delta}{۲} < \frac{\delta^*}{۲}$$

. با توجه با لم ۴ــ۱۷، $c_p \in B_1^* \cap B_2^*$ است. اگر $c_p \notin B_2^*$ باشد، در نتیجه $c_p \in B_1^*$ است. و در نتیجه با توجه به مشاهده‌ی ۴ــ۱۳، B_2 حداکثر یکی از B_1^* و B_2^* را قطع می‌کند و در نتیجه $B_2 \cap B_2^* = \emptyset$.

□

الگوریتم پاسخ‌گویی به پرسمان، به ازای هر نقطه‌ی $c \in C$، دو توپ $B_1'(c_1, \max\left\{\delta, \frac{۲}{\alpha}\|c_1c\|\right\})$ و $B_2'(c, \frac{۲}{\alpha}\|c_1c\|)$ را تشکیل می‌دهد. اگر نقطه‌ی کاندید برابر c_2 کنونی باشد، $B_1' = B_1$ است. از طرفی چون $\frac{\delta}{۲} \geqslant r \geqslant \frac{۲}{\alpha}\|c_1c_2\|$ است، با توجه به ناوردای ۲، $B_2' \subset B_2$ است و در نتیجه نیازی به ساخت B_1' و B_2' نیست. اگر نقطه‌ی کاندید برابر c_2 نباشد، با توجه به ناوردای ۲، مطمئن هستیم که $B_1 \subset B_1'$ است. بنابراین کافی است ببینم کدام‌یک از نقاط $B_1 \cup Buffer$ داخل B_1' قرار می‌گیرند.

اگر $|B_2| \geqslant (d+۱)(z+۱)$ باشد، بدان معناست که $c_p \notin B_2^*$ است. با توجه به لم ۴ــ۲۰، 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 $AddToB_1'$ و $AddToB_2'$ به ترتیب برای اضافه کردن نقاط به B_1' و B_2' استفاده می‌کند. این دو تابع، کاملا مشابه $AddToB_1$ و $AddToB_2$ 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 کاندیداها را بررسی می‌کند، حداقل به ازای یک $c^* \in C$، $c^* \in B_2^*$ نیز قرار دارد. B'_1 و B'_2 متناظر با نقطه‌ی کاندید c^* را به ترتیب B''_1 و B''_2 نشان می‌دهیم. با توجه به مشاهده‌ی ۴ــ۱۲ داریم:

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

و از طرفی داریم:

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

و در نتیجه با توجه به لم ۴ــ۱۶ $B_1^* \subset B''_1$ و $B_2^* \subset B''_2$ است. که در واقع بدان معناست که B''_1 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.

قضیه‌ی ۴ــ۲۱ *الگوریتم ۱۰ ، در حالتی که دو توپ بهینه αــ جداپذیر باشند، می‌تواند جوابی با ضریب تقریب $1/8 + \epsilon$ برای مسئله‌ی ۲ــ 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 $O(zd) \times O(\frac{z^2 d}{\epsilon})$ 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 (فرض شده نقاط اول هر دسته داده‌ی پرت نیست). 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 ۴ــ۲۲ *در حالتی که $\delta^* > \alpha r^*$ است، یک $1/8 + \epsilon$ــ تقریب برای مسئله‌ی ۲ــ مرکز با z داده‌ی پرت قابل نگه‌داری است که از مرتبه‌ی $O(\frac{z^3 d}{\epsilon})$ حافظه‌ی مصرف کرده و زمان به‌روزرسانی آن، از مرتبه‌ی $O(\frac{z^5 d^2}{\epsilon})$ بوده و با مصرف $O(\frac{z^5 d^3}{\epsilon})$ 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 جویبارداده، 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 βــ تقریب برای مسئله‌ی ۱ــ 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 نگه می‌داریم که به طور موازی نگه‌داری می‌شود. توجه کنید که داده‌ساختاره‌های 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 شده است.*

برای نگه‌داری B_1 و B_2 و B_u از الگوریتم جویبارداده‌ی ۵ ارائه شده در همین پایان‌نامه استفاده می‌کنیم، که یک الگوریتم با ضریب تقریب $1/8+\epsilon$ ارائه می‌دهد. با توجه به الگوریتم ۵، حافظه‌ی مصرفی برابر $\mathcal{O}(\frac{zd}{\epsilon}) \times \mathcal{O}(z)$ 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 $\mathcal{O}(\frac{dz^2}{\epsilon}) \times \mathcal{O}(z) \times \mathcal{O}(zd)$ زمان می‌برد. از طرفی با توجه به این‌که برای حذف فرض پرت نبودن نقطه‌ی اول نیاز به اجرای z نمونه موازی ار الگوریتم داریم، بنابراین تمام تحلیل‌ها z برابر می‌شوند.

□

فصل ۵

نتیجه‌گیری

در این فصل، ضمن جمع‌بندی نتایج جدید ارائه‌شده در پایان‌نامه، مسائل باز باقی‌مانده و همچنین پیش‌نهادهایی برای ادامه‌ی کار ارائه می‌شوند.

کتاب‌نامه

[1] *S.-S. Kim and H.-K. Ahn. An improved data stream algorithm for clustering. In* Proceedings of the 11th*, pages 273–284. 2014.*

[2] *J. Han and M. Kamber.* Data Mining, Southeast Asia Edition: Concepts and Techniques*. Morgan kaufmann, 2006.*

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

[4] *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.*

[5] *R. M. McCutchen and S. Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In* Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 165–178. 2008.*

[6] *M. Sipser.* Introduction to the Theory of Computation*. Cengage Learning, 2012.*

[7] *I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover.* Annals of mathematics*, pages 439–485, 2005.*

[8] *V. V. Vazirani.* Approximation algorithms*. Springer Science & Business Media, 2013.*

[9] *M. Bern and D. Eppstein. Approximation algorithms for geometric problems.* Approximation algorithms for NP-hard problems*, pages 296–345, 1996.*

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

[12] *R. G. Michael and S. J. David. Computers and intractability: a guide to the theory of np-completeness.* WH Freeman & Co., San Francisco*, 1979.*

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

[14] *J. Han, M. Kamber, and J. Pei.* Data mining: concepts and techniques: concepts and techniques*. Elsevier, 2011.*

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

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

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

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

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

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

[21] *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.*

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

[23] *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.*

[24] *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.*

[25] *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.*

[26] *M. Bâdoiu and K. L. Clarkson. Smaller core-sets for balls. In* Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms*, pages 801–802. Society for Industrial and Applied Mathematics, 2003.*

[27] *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.*

[28] *L. Danzer, B. Gruenbaum, and V. Klee. Helly's theorem and its relatives. In* Proceedings of the Symposia in Pure Mathematics 7*, pages 101–180, 1963.*

واژه‌نامه

الف

ابتکاری .... heuristic

ارزش .... worth

Abstract

This part should be completed.

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-zade

September 2015