نشریه علمی‌ترویجی محاسبات نرم

سال ششم/ شمارهٔ اول/ بهار و تابستان ۱۳۹۶/ صفحهٔ ۲ـ۳۷

دریافت مقاله: ۱۳۹۳/۹/۷

پذیرش مقاله: ۱۳۹۶/۹/۲

بررسی مشکلات الگوریتم خوشه‌بندی DBSCAN و مروری بر بهبودهای ارائه‌شده برای آن

علی زاده ده‌بالایی[۱*]، علیرضا باقری[۲]، حامد افشار[۳]

[۱] کارشناس ارشد، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، ایران
alizadehei@aut.ac.ir

[۲] استادیار، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، ایران
ar_bagheri@aut.ac.ir

[۲] کارشناس ارشد، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، ایران
hafshar@aut.ac.ir

چکیده

خوشه‌بندی یک از تکنیک‌های مهم کشف دانش در پایگاه داده است. الگوریتم‌های خوشه‌بندی مبتنی بر چگالی یکی از روش‌های اصلی برای خوشه‌بندی در داده‌کاوی هستند. عدم محدودیت به شکل خوشه‌ها، ساده و قابل‌فهم بودن از جمله مزایای این الگوریتم‌ها است. DBSCAN الگوریتم پایهٔ روش‌های خوشه‌بندی مبتنی بر چگالی است. این الگوریتم قابلیت کشف خوشه‌های با اندازه و اشکال متفاوت را از حجم زیادی از داده‌ها دارد و در مقابل نویز نیز مقاوم است. علی‌رغم وجود این مزایا، این الگوریتم دارای مشکلاتی نظیر سخت بودن تعیین مقدار دقیق پارامترهای ورودی، عدم‌تشخیص خوشه‌های با چگالی متفاوت و عدم‌تشخیص صحیح خوشه‌ها در هنگام نزدیک بودن خوشه‌ها به هم نیز می‌باشد.

از سال ۱۹۹۶ که DBSCAN ارائه شده تا به امروز، الگوریتم‌های بسیار زیادی در جهت بهبود DBSCAN ارائه شده‌اند. در این مقاله ابتدا، مشکلات الگوریتم DBSCAN بررسی می‌شوند. سپس به مرور و بررسی الگوریتم‌هایی که در جهت بهبود مشکلات الگوریتم DBSCAN ارائه شده‌اند می‌پردازیم تا با نقاط ضعف و قوت این الگوریتم‌ها و میزان موفقیت این الگوریتم‌ها در بهبود الگوریتم DBSCAN آشنا شویم. همچنین، با توجه به مطالعات انجام‌شده، اقدام به پیاده‌سازی برخی از این الگوریتم‌ها نموده‌ایم و آن‌ها را بر روی مجموعه داده‌های استاندارد، بر اساس معیارهای ارزیابی خوشه‌بندی تست کرده‌ایم تا بهتر بتوانیم دربارهٔ این الگوریتم‌ها قضاوت کنیم.

واژه‌های کلیدی: خوشه‌بندی مکانی، DBSCAN، مبتنی بر چگالی، چگالی متفاوت، تعیین پارامتر، پایگاه داده مکانی.

۱. مقدمه

سیستم مدیریت پایگاه داده مکانی [۱]، یک سیستم پایگاه داده جهت مدیریت مؤثر و کارآمد انواع دادهٔ مکانی در فضاهای دوبعدی، سه‌بعدی یا حتی فضاهای با بعد بالاتر است. این سیستم در زمینه‌هایی از جمله داده‌های مکانی جغرافیایی و دانش پزشکی به‌شدت در حال استفاده است. با توجه به حجم بسیار بالا داده‌های به‌دست‌آمده از تصاویر ماهواره‌ای و سایر تجهیزات خودکار و عدم توانایی انسان در تجزیه‌وتحلیل این حجم بالا از داده‌ها، نیاز داریم تا روش‌های کشف دانش مانند کاوش دانش در داده‌های مکانی را بر روی داده‌ها اعمال کنیم. داده‌کاوی مکانی [۲] عبارت است از استخراج دانش و روابط مکانی و هر خصوصیت دیگری که به‌صورت غیر ضمنی در پایگاه داده ذخیره شده است. داده‌کاوی مکانی برای پیدا کردن قواعد و روابط ضمنی بین داده‌های مکانی و غیرمکانی مورد استفاده قرار می‌گیرد. تلاش‌های تحقیقاتی که در زمینهٔ داده‌کاوی مکانی صورت گرفته است، در زیر چترهای مختلفی مانند آمار مکانی و محاسبات جغرافیایی قرار گرفته‌اند.

خوشه‌بندی، یعنی گروه‌بندی اشیاء پایگاه داده در زیرگروه‌های معنادار، یکی از کارهای مهم کشف دانش در پایگاه داده مکانی است [۳]. تجزیه‌وتحلیل خوشه به‌طور گسترده‌ای در تجزیه‌وتحلیل داده‌ها مورد استفاده قرار می‌گیرد، به این صورت که یک مجموعه از اقلام داده‌ای را به داخل گروه‌ها یا خوشه‌هایی سازمان‌دهی می‌کند، به‌طوری که اقلامی که در داخل یک خوشه هستند، دارای خصوصیات مشابه به هم‌اند و با اقلام موجود در سایر خوشه‌ها متفاوت‌اند [۴]. تحقیقات بسیار زیادی در زمینه خوشه‌بندی داده‌ها صورت گرفته است. به‌طور کلی یک الگوریتم خوشه‌بندی باید دارای ویژگی‌های زیر باشد:

۱. با توجه به اینکه در پایگاه داده‌های بزرگ تعیین مقدار دقیق برای پارامترهای ورودی الگوریتم‌های خوشه‌بندی مشکل است، الگوریتم‌ها باید به‌گونه‌ای طراحی شوند که حداقل نیازمندی به دانش دامنه‌ای برای تعیین پارامترهای ورودی را داشته باشند.
۲. الگوریتم‌های خوشه‌بندی باید قابلیت کشف خوشه‌های با اشکال اختیاری را داشته باشند؛ چراکه ممکن است شکل خوشه‌ها در پایگاه داده‌های مکانی غیر محدب، کروی و به‌طوری کلی هر شکل هندسی باشد.
۳. الگوریتم‌های خوشه‌بندی باید در پایگاه داده‌های بسیار بزرگ (بیش از ۱۰۰۰ شیء) کارایی داشته باشند.

تعداد زیادی از الگوریتم‌های خوشه‌بندی تا این زمان به‌منظور مرتفع ساختن این نیازمندی‌ها معرفی شده‌اند. تا به امروز هیچ‌گونه الگوریتم واحدی که قابلیت رسیدگی به همهٔ نیازمندی‌های گفته‌شده را داشته باشد، ارائه نشده است.

روش‌های بسیاری برای انجام خوشه‌بندی پیشنهاد شده است که این روش‌ها را می‌توان در پنج نوع اصلی دسته‌بندی کرد: پارتیشن‌بندی، مبتنی بر سلسله‌مراتب، مبتنی بر چگالی، مبتنی بر مدل و مبتنی بر شبکه. جزئیات مربوط به دسته‌بندی الگوریتم‌های خوشه‌بندی را می‌توان در [۵] مشاهده کرد. الگوریتم‌های خوشه‌بندی مبتنی بر چگالی یکی از روش‌های اصلی برای خوشه‌بندی در داده‌کاوی هستند. روش‌های مبتنی بر چگالی بر مبنای مفهوم چگالی توسعه داده شده‌اند. خوشه‌های مبتنی بر چگالی به‌عنوان خوشه‌هایی تعریف می‌شوند که به‌وسیلهٔ چگالی‌های مختلف از سایر خوشه‌ها متفاوت هستند. از جمله مزایای الگوریتم‌های خوشه‌بندی مبتنی بر چگالی می‌توان به موارد زیر اشاره کرد:

- سادگی و قابل‌فهم بودن
- عدم نیاز به تعیین تعداد خوشه‌های موجود در مجموعه داده (برخلاف الگوریتم‌های پارتیشن‌بندی)
- قابلیت تشخیص خوشه‌های با اشکال اختیاری
- سادگی تشخیص و حذف نویز

الگوریتم‌های خوشه‌بندی مبتنی بر چگالی بسیاری از جمله DBSCAN [۶]، DENCLUE [۷]، DBCLASD [۸] و OPTICS [۹] ارائه شده است. DBSCAN الگوریتم پایهٔ روش‌های خوشه‌بندی مبتنی بر چگالی است. این الگوریتم قابلیت کشف خوشه‌های با اندازه و اشکال متفاوت را از حجم زیادی از داده‌ها دارد و در مقابل نویز نیز مقاوم است. علی‌رغم

وجود این مزایا، این الگوریتم کمبودهایی نیز دارد. اول اینکه این الگوریتم نیاز به دو پارامتر ورودی MinPts و Eps دارد که تعیین این پارامترها به‌خصوص در پایگاه داده‌های با حجم بالا بسیار سخت است. دوم اینکه این الگوریتم قابلیت کشف خوشه‌های با چگالی متفاوت را ندارد. همچنین، DBSCAN در مواقعی که خوشه‌ها نزدیک به هم باشند، در تشخیص صحیح خوشه‌ها ممکن است با شکست مواجه شود. علاوه بر این، پیچیدگی زمانی بالای این الگوریتم باعث شده تا این الگوریتم در مواجهه با پایگاه داده‌های بسیار حجیم و داده‌های با بعد بالا کارایی چندان مناسبی نداشته باشد.

از سال ۱۹۹۶ که الگوریتم DBSCAN ارائه شد تا به امروز، تلاش‌های بسیاری در جهت مرتفع ساختن این مشکلات و بهبود الگوریتم DBSCAN صورت گرفته است. این موضوع باعث ایجاد یک چالش بزرگ برای کاربران شده است، به‌گونه‌ای که کاربران بایستی از بین روش‌های موجود یکی را انتخاب کنند. ازاین‌رو داشتن آگاهی کافی در زمینهٔ روش‌های مختلف ارائه شده و آشنایی با نقاط ضعف و قوت آن‌ها به ما در انتخاب الگوریتم موردنظر کمک خواهد کرد. هدف اصلی ما در این مقاله، بررسی مشکلات الگوریتم DBSCAN و معرفی و بررسی نقاط ضعف و قوت بهبودهای ارائه‌شده در جهت رفع این مشکلات و برخی از بهبودهای دیگر الگوریتم DBSCAN است. یکی از مشکلات اساسی الگوریتم DBSCAN، عدم توانایی این الگوریتم در تشخیص خوشه‌های با چگالی متفاوت است. این در حالی است که یکی از خصوصیات مهم مجموعه داده‌های دنیای واقعی این است که خوشه‌های موجود در این مجموعه داده‌ها، به‌دلیل وجود چگالی‌های محلی متفاوت، فقط با یک تنظیم پارامتر سراسری قابل‌تشخیص نیستند. ازاین‌رو، ما در این مقاله این مشکل را بیشتر مورد بررسی قرار داده‌ایم، به‌گونه‌ای که با انتخاب چند تا از بهترین الگوریتم‌های ارائه‌شده برای رفع این مشکل و پیاده‌سازی این الگوریتم‌ها، اقدام به مقایسهٔ این الگوریتم‌ها بر اساس معیارهای ارزیابی روش‌های خوشه‌بندی نموده‌ایم و نهایتاً بهترین این الگوریتم‌ها را معرفی کرده‌ایم.

بقیهٔ مقاله در بخش‌های زیر سازمان‌دهی شده است. در قسمت ۳، به معرفی برخی از کاربردهای الگوریتم‌های خوشه‌بندی می‌پردازیم. در قسمت ۴ ما الگوریتم DBSCAN را مختصراً شرح داده و سپس در قسمت ۵ به بررسی مشکلات این الگوریتم می‌پردازیم. در قسمت ۶ نیز بهبودهای ارائه‌شده برای الگوریتم DBSCAN را معرفی و بررسی می‌کنیم. در قسمت ۷ به مقایسهٔ بین برخی از الگوریتم‌ها می‌پردازیم. در پایان نیز نتیجه‌گیری ارائه شده است.

۲. کارهای مرتبط

در این قسمت قصد داریم تا دربارهٔ مقاله‌های مروری موجود بر روی روش‌های خوشه‌بندی و تفاوت این مقاله‌ها با مقاله فعلی صحبت کنیم. مقاله‌های مروری موجود را می‌توانیم در قالب دو دسته بررسی کنیم:

۱. مقاله‌های مروری که به بررسی همهٔ روش‌های خوشه‌بندی پرداخته‌اند.
۲. مقاله‌های مروری که به‌طور خاص به بررسی روش‌های خوشه‌بندی مبتنی بر چگالی پرداخته‌اند.

همان طور که قبلاً نیز اشاره کردیم، به‌طور کلی الگوریتم‌های خوشه‌بندی به پنج دستهٔ پارتیشن‌بندی، مبتنی بر سلسله‌مراتب، مبتنی بر چگالی، مبتنی بر مدل و مبتنی بر شبکه تقسیم‌بندی می‌شوند. اکثر مقاله‌های مروری موجود که به بررسی الگوریتم‌های خوشه‌بندی پرداخته‌اند، بر روی همهٔ روش‌های خوشه‌بندی موجود در این پنج دسته متمرکز هستند. از جمله این مقاله‌ها می‌توان به [۱۰، ۱۱ و ۱۲] اشاره کرد. این گونه مقالات مروری به‌طور خاص بر روی بهبودهای الگوریتم DBSCAN متمرکز نیستند، به گونه‌ای که در این مقاله‌ها فقط گاهاً به چند مورد از الگوریتم‌های معروف و پایه‌ای روش‌های خوشه‌بندی مبتنی بر چگالی مانند الگوریتم OPTICS [۹] اشاره شده است. این در حالی است که مقالهٔ فعلی به‌طور خاص به بررسی الگوریتم خوشه‌بندی DBSCAN و بهبودهای آن پرداخته است.

در رابطه با مقاله‌های مروری که به بررسی الگوریتم‌های خوشه‌بندی مبتنی بر چگالی و به‌طور خاص به بررسی بهبودهای الگوریتم DBSCAN پرداخته‌اند، دو نکته وجود دارد: اول اینکه تعداد این مقاله‌ها بسیار کم است؛ دوم اینکه این مقاله‌ها اطلاعات بسیار محدودی در اختیار خواننده قرار داده‌اند و فقط به بررسی تعداد معدودی الگوریتم خوشه‌بندی پرداخته‌اند. برای مثال، در [۱۳] به بررسی فقط پنج روش خوشه‌بندی مبتنی بر چگالی پرداخته شده است. این در حالی است که در مقالهٔ فعلی، حدود ۲۶ الگوریتم که برای بهبود الگوریتم DBSCAN ارائه شده‌اند، معرفی و بررسی شده‌اند. بنابراین، هیچ مقاله‌ای به اندازهٔ مقاله فعلی جامعیت ندارد.

این مقاله از ساختار بسیار خوبی برخوردار است. در ابتدا الگوریتم DBSCAN و کاربردها و مزایای این الگوریتم به‌طور کامل شرح داده شده است. سپس با توجه به بررسی‌های صورت‌گرفته، چهار مشکل اساسی الگوریتم DBSCAN به‌طور کامل شرح داده شده است. بعد از آن گفته شده که برای هر مشکل این الگوریتم چه بهبودهایی ارائه شده است. در بخش معرفی بهبودهای این الگوریتم نیز سعی شده به‌طور خلاصه، نحوهٔ عملکرد این الگوریتم‌ها شرح داده شود و مزایا، معایب و کارهای آتی که می‌توان بر روی این الگوریتم‌ها ارائه داد مشخص شده است. در پایان مقاله نیز در قالب یک جدول به‌طور خلاصه هر ۲۶ الگوریتم با یکدیگر مقایسه شده‌اند.

در این مقاله، مشکل اساسی الگوریتم DBSCAN یعنی عدم توانایی تشخیص خوشه‌های با چگالی متفاوت، بیشتر مورد بررسی قرار گرفته است، به‌گونه‌ای که با انتخاب چند تا از بهترین الگوریتم‌های ارائه‌شده برای رفع این مشکل و پیاده‌سازی این الگوریتم‌ها، اقدام به مقایسهٔ این الگوریتم‌ها بر اساس معیارهای ارزیابی روش‌های خوشه‌بندی و با استفاده از مجموعه‌داده‌های استاندارد نموده‌ایم و در نهایت بهترین این الگوریتم‌ها را معرفی کرده‌ایم. این مقایسه همچنین در راستای هدف مقاله، جهت آشنایی خوانندگان با چگونگی مقایسه الگوریتم‌های خوشه‌بندی نیز می‌باشد.

علاوه بر این با پیاده‌سازی و آزمایش برخی از این الگوریتم‌ها به بررسی دقیق‌تر نقاط قوت و ضعف آن‌ها پرداخته‌ایم. برای مثال با پیاده‌سازی و بررسی نتایج الگوریتم VDBSCAN به اشکالات اساسی این الگوریتم پی برده‌ایم. در ادامه مقاله بیشتر در این مورد صحبت خواهیم کرد.

به‌طور خلاصه، این مقاله شامل قسمت‌های زیر می‌باشد که در واقع وجه تمایز این مقاله با سایر مقاله‌های مروری در این زمینه است و هیچ مقاله‌ای با چنین ساختار و محتوایی و تا این حد کامل اطلاعات در اختیار خوانندگان قرار نداده است:

۱. معرفی و شرح کامل الگوریتم DBSCAN
۲. بیان کاربردهای الگوریتم DBSCAN
۳. بیان مزایا و معایب الگوریتم DBSCAN
۴. معرفی الگوریتم‌هایی که برای رفع مشکلات الگوریتم DBSCAN ارائه شده‌اند شامل شرح الگوریتم، بیان مزایا و معایب (درصورت وجود) الگوریتم و کارهای آتی که برای هر الگوریتم قابل انجام است.
۵. پیاده‌سازی پنج الگوریتم که برای رفع یکی از مشکلات الگوریتم DBSCAN ارائه شده‌اند و مقایسهٔ آن‌ها بر روی مجموعه داده‌های استاندارد و بر اساس معیارهای استاندارد ارزیابی روش‌های خوشه‌بندی و نهایتاً معرفی بهترین این الگوریتم‌ها بر اساس نتایج آزمایش‌ها.
۶. مقایسهٔ نسخه‌های بهبودی الگوریتم DBSCAN در قالب یک جدول.
۷. ارائهٔ پیشنهاداتی در انتهای مقاله برای کارهایی که در زمینه الگوریتم‌های خوشه‌بندی مبتنی برچگالی در آینده قابل انجام است.

۳. کاربرد الگوریتم‌های خوشه‌بندی

خوشه‌بندی به‌عنوان یک روش یادگیری بدون نظارت دارای کاربردهای متعدد در زمینه‌های مختلف است. به‌طور کلی از جمله کاربردهای الگوریتم‌های خوشه‌بندی می‌توان به موارد زیر اشاره کرد:

- موتورهای جستجو [۱۴]: الگوریتم‌های خوشه‌بندی را می‌توان به‌عنوان ستون فقرات موتورهای جستجوگر در نظر

گرفت. موتورهای جستجو سعی می‌کنند اشیاء شبیه به هم را در داخل یک خوشه و اشیاء بی‌شباهت را در داخل یک خوشهٔ دیگر قرار دهند. نتایج حاصل از خوشه‌بندی می‌توانند برای قرار گرفتن در صفحهٔ اول نتایج جستجو مورد استفاده قرار گیرند.

- جامعهٔ دانشگاهی [۱۵]: توانایی نظارت بر پیشرفت عملکرد تحصیلی دانش‌آموزان یک موضوع مهم برای جامعهٔ دانشگاهی است. الگوریتم‌های خوشه‌بندی را می‌توان برای نظارت بر عملکرد تحصیلی دانش‌آموزان استفاده کرد. در واقع بر اساس نمرات دانش‌آموزان، آن‌ها را به گروه‌های مختلف تقسیم می‌کنیم که هر گروه نشان‌دهندهٔ یک سطح از عملکرد است.
- برنامه‌های مبتنی بر حسگرهای شبکه بی‌سیم [۱۶]: یکی دیگر از کاربردهای الگوریتم‌های خوشه‌بندی، برنامه‌های مبتنی بر حسگرهای شبکه بی‌سیم است. برای مثال یکی از برنامه‌هایی که می‌توان در آن از الگوریتم‌های خوشه‌بندی استفاده کرد، برنامهٔ تشخیص مین‌های زمینی است که در این برنامه‌ها الگوریتم‌های خوشه‌بندی نقش پیدا کردن مراکز خوشه‌ها را دارند.
- خوشه‌بندی نواحی جغرافیایی بر اساس معیارهای مختلف: از جمله دیگر کاربردهای الگوریتم‌های خوشه‌بندی، خوشه‌بندی نواحی جغرافیایی بر اساس معیارهای مختلف است. برای مثال در [۱۷] از الگوریتم K-Means [۱۸] که یک الگوریتم مبتنی بر پارتیشن است، برای خوشه‌بندی استان‌های ایران بر پایهٔ معیارهای شکاف دیجیتال استفاده شده است.
- حاشیه‌نویسی تصاویر: یکی دیگر از کاربردهای الگوریتم‌های خوشه‌بندی، حاشیه‌نویسی تصاویر است. با استفاده از حاشیه‌نویسی تصویر، کلماتی که بیانگر معنا و مفهوم واقعی تصاویرند، با تصویر همراه می‌شوند. امروزه با توجه به حجم روزافزون تصاویر دیجیتال، تجزیه‌وتحلیل این حجم از تصاویر توسط انسان دشوار، پرهزینه و زمانبر است. ازاین‌رو، نیاز به استفاده از روش‌هایی برای خودکارسازی حاشیه‌نویسی تصاویر داریم. در [۱۹] از الگوریتم‌های خوشه‌بندی نیم‌نظارتی طیفی[1] برای حاشیه‌نویسی تصاویر استفاده شده است.

داده‌کاوی، بازاریابی، زیست‌شناسی، نقشه‌برداری شهری، مطالعات زلزله‌نگاری، کتابداری، وب، تشخیص گفتار و تقسیم‌بندی تصاویر از جمله دیگر کاربردهای الگوریتم‌های خوشه‌بندی هستند [۲۰].

با توجه به اینکه این مقاله بر روی الگوریتم DBSCAN و توسعه‌های آن تمرکز دارد، در ادامه برخی از کاربردهای مربوط به الگوریتم DBSCAN را بیان می‌کنیم:

- تصاویر ماهواره‌ای: همه‌روزه داده‌های بسیار زیادی از ماهواره‌ها دریافت می‌شود. این داده‌ها باید به اطلاعات قابل‌فهمی تبدیل شوند. طبقه‌بندی نواحی موجود در تصاویر گرفته‌شده توسط ماهواره‌ها بر اساس جنگل‌ها، آب‌ها و کوه‌ها نمونه‌ای از این تبدیل اطلاعات است. قبل از اینکه الگوریتم DBSCAN این اطلاعات را طبقه‌بندی کند، نیاز است تا بر روی تصاویر عمل پردازش تصویر انجام شود. بعد از انجام پردازش تصویر، داده‌ها در قالب داده‌های مکانی ظاهر می‌شوند و الگوریتم DBSCAN می‌تواند عمل طبقه‌بندی را انجام دهد.
- کریستالوگرافی اشعه X: کریستالوگرافی اشعه X یکی دیگر از کاربردهای عملی الگوریتم DBSCAN است. در اینجا الگوریتم DBSCAN می‌تواند برای پیدا کردن و طبقه‌بندی اتم‌های موجود در داده‌ها مورد استفاده قرار بگیرد.
- تشخیص ناهنجاری در اطلاعات دما: این نوع از برنامه‌ها بر روی الگوهای ناهنجاری موجود در داده‌ها تمرکز دارند. در اینجا هدف یافتن ناهنجاری‌های موجود در اطلاعات دماست [۲۱] که مرتبط با تغییرات محیطی هستند. این ناهنجاری‌ها معمولاً به‌دلیل خطاهای موجود در تجهیزات مربوطه ایجاد می‌شوند. این الگوهای نامعمول نیاز به شناسایی و بررسی دارند. الگوریتم

1. Semi-supervised spectral clustering

خوشه‌بندی DBSCAN قابلیت کشف اینچنین الگوهایی را از داده‌ها دارد.

۴. الگوریتم DBSCAN

الگوریتم DBSCAN نیاز به دو پارامتر Minpts و Eps دارد. در واقع با استفاده از این دو پارامتر ما حداقل چگالی یک خوشه را تعیین می‌کنیم. برای درک فرایند این الگوریتم نیاز است که برخی از تعاریف پایه را معرفی کنیم.

۱.۴. تعاریف اولیه

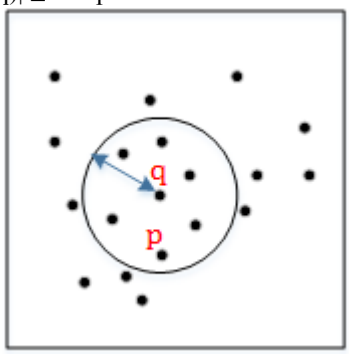
تعریف ۱: همسایه‌های شعاع Eps یک نقطه [۶]:همسایه‌های موجود در شعاع Eps یک نقطه مثل p که با NEps(p) نشان داده می‌شوند، مجموعه‌ای از نقاط هستند که فاصله‌شان از P کمتر از شعاع Eps باشد، یعنی:

$$NEps(p)=\{q\epsilon D \mid Dist(p,q)\leq Eps\} \quad (۱)$$

تعریف ۲: شیء مرکزی [۶]: به شیئی که حداقل تعداد Minpts شیء در شعاع همسایگی Eps خود داشته باشد، شیء مرکزی گفته می‌شود.

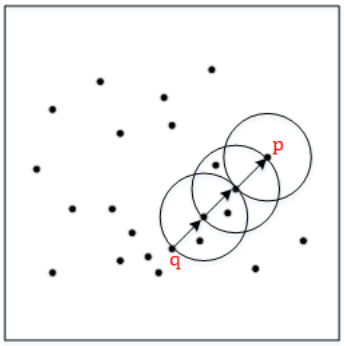
تعریف ۳: دسترسی‌پذیر چگالی مستقیم [۶]: همان طور که در شکل (۱) می‌بینید، نقطهٔ p دسترسی‌پذیر چگالی مستقیم از نقطهٔ q است، اگر، اولاً p جزء همسایه‌های شیء q باشد و ثانیاً شیء q یک شیء مرکزی باشد، یعنی:

1. $p \epsilon NEps(q)$
2. $|NEps(q)| \geq Minpts$



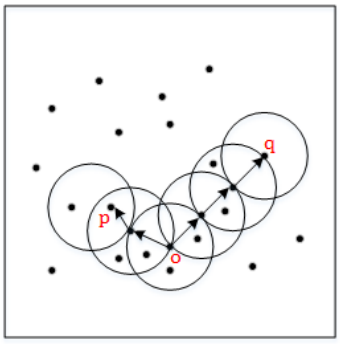
شکل (۱): شیء p دسترسی‌پذیر چگالی مستقیم از شیء q است.

تعریف ۴: دسترسی‌پذیر چگالی [۶]: نقطهٔ p دسترسی‌پذیر چگالی از نقطهٔ q است، اگر یک زنجیره از نقاط $p_1,p_2,p_3,\ldots,p_n$ که $p_1=q$ و $p_n=p$ وجود داشته باشد، به‌گونه‌ای که p_{i+1} دسترسی‌پذیر چگالی مستقیم از P_i باشد. شکل (۲) این مفهوم را نشان می‌دهد.



شکل (۲): شیء p دسترسی‌پذیر چگالی از شیء q است.

تعریف ۵: متصل چگالی [۶]: نقطهٔ p متصل چگالی از نقطه q است اگر، همانند شکل (۳) یک نقطه مثل o وجود داشته باشد به‌گونه‌ای که هردوی p و q دسترسی‌پذیر چگالی از o باشند.



شکل (۳): شیء p متصل چگالی از شیء q است.

تعریف ۶: خوشه [۶]: فرض کنید که D یک پایگاه داده از نقاط باشد. خوشهٔ C یک زیرمجموعه غیر تهی از D است، به‌گونه‌ای که شرط‌های زیر را ارضاء کند:

۱. به‌ازای همه جفت نقاط p و q، اگر $p\epsilon C$، یعنی p یکی از اعضای خوشهٔ C باشد، و همچنین q نیز دسترسی‌پذیر چگالی از p باشد، آنگاه q نیز باید متعلق به خوشهٔ C باشد. (شرط حداکثر بودن)

۲. به‌ازای همه جفت نقاط p و q متعلق به خوشهٔ C، p باید متصل چگالی از q باشد. (شرط اتصال)

تعریف ۷: نویز [۶]: فرض کنید که $C_1,C_2,..,C_k$ خوشه‌های یافت‌شده از پایگاه داده D باشند. به مجموعه‌ای از

نقاط که در پایگاه داده D وجود دارند ولی متعلق به هیچ‌یک از خوشه‌های $C_i \epsilon 1 \text{ to } k$ نباشند، نویز می‌گویند:

$$Noise=\{p \epsilon D \mid \forall_i : p \notin C_i\} \quad (۲)$$

تعریف ۸: شیء حاشیه‌ای [۶]: شیء حاشیه‌ای به شیئی گفته می‌شود که شیء مرکزی نباشد، منتها از یک شیء مرکزی دیگر دسترسی‌پذیر چگالی باشد.

با توجه به تعاریف اولیهٔ ارائه‌شده در این قسمت، در قسمت بعد الگوریتم DBSCAN را شرح می‌دهیم.

۲.۴. الگوریتم DBSCAN

شبه کد مربوط به الگوریتم DBSCAN در شکل (۴) نشان داده شده است.

Algorithm #1: DBSCAN Algorithm
Input: D, Minpts, Eps

```
1. C=0
2. For each unvisited point p in the dataset D
3.   Mark p as visited
4.   N=regionQuery(p,Eps)
5.   If sizeof(N)< Minpts
6.     Mark p as Noise
7.   Else
8.     Put p into new cluster C
9.     Enlargecluster(Eps,Minps,C,p,N)
10.  End If
11. End For
12. End // DBSCAN
// Enlargecluster Function
Enlargecluster(Eps,Minpts,C,p,N)
13. Add p to cluster C
14. For each point p' in N
15.   If p' is unvisited
16.     Mark p' as visited
17.     N'=regionQuery(p',Eps)
18.     If sizeof(N')>= Minpts
19.       N=N' combine to N
20.     End If
21.     If p' is not in any cluster
22.       Add p' to cluster C
23.   End If
24.  End If
25. End For
26. End // Enlargecluster
```

شکل (۴): الگوریتم DBSCAN [۲۲]

الگوریتم با یک نقطهٔ اختیاری p از مجموعه داده شروع می‌کند و همهٔ اشیاء دسترسی‌پذیر چگالی از آن نقطه را بازیابی می‌کند. اگر تعداد همسایه‌های شعاع Eps نقطهٔ p کمتر از Minpts بود؛ یعنی p یک نقطهٔ مرکزی نبود، الگوریتم p را به‌عنوان نویز برچسب می‌زند و سراغ نقطهٔ بعدی از پایگاه داده می‌رود. در غیر این صورت، یعنی اگر p یک نقطهٔ مرکزی باشد، یک خوشهٔ جدید تشکیل می‌شود و نقطهٔ p برچسب این خوشه را می‌گیرد. سپس نقطهٔ p و همسایه‌های شعاع Eps آن (لیست N) به تابع Enlargecluster جهت بسط بیشتر فرستاده می‌شوند. در تابع Enlargecluster به‌ازای هرکدام از همسایه‌های شعاع Eps نقطهٔ p مانند p'، در صورتی که نقطهٔ p' ملاقات نشده باشد و یک نقطهٔ مرکزی باشد، بسط داده می‌شود و همسایه‌های آن به لیست N جهت بسط بیشتر افزوده می‌شوند. در غیر این صورت p' به‌طور ساده به خوشه افزوده می‌شود و دیگر بسط داده نمی‌شود. این فرایند تا زمانی ادامه دارد که لیست N خالی شود. سپس الگوریتم سراغ نقطهٔ بعدی از پایگاه داده که هنوز ملاقات نشده است می‌رود و کار را به همین طریق ادامه می‌دهد تا تمامی نقاط پایگاه داده ملاقات شوند.

۳.۴. پیچیدگی زمانی الگوریتم DBSCAN

حال اجازه بدهید تا دربارهٔ پیچیدگی زمانی این الگوریتم صحبت کنیم. بخش زمان‌بر این الگوریتم پرس‌وجوی ناحیه‌ای است که به‌ازای هر نقطه برای پیدا کردن همسایه‌های آن نقطه به کار می‌رود. در صورتی که از ساختارهای شاخص مکانی استفاده نکنیم، این پرس‌وجو از مرتبهٔ O(n) خواهد بود؛ یعنی برای پیدا کردن همسایه‌های دسترسی‌پذیر چگالی یک نقطه مجبور هستیم تا همهٔ نقاط مجموعه داده را بررسی کنیم. این پرس‌وجو برای همهٔ نقاط انجام می‌شود. بنابراین پیچیدگی زمانی این الگوریتم برابر با $O(n^2)$ خواهد بود. اما در صورت استفاده از ساختارهای شاخص مکانی مانند R*-tree، هر پرس‌وجوی ناحیه‌ای حداکثر به‌اندازهٔ ارتفاع R*-tree یعنی O(logn) طول خواهد کشید. بنابراین پیچیدگی زمانی الگوریتم DBSCAN با استفاده از ساختارهای شاخص مکانی از مرتبه O(nlogn) است.

۵. بررسی مشکلات الگوریتم DBSCAN

همان گونه که قبلاً نیز اشاره کردیم، الگوریتم خوشه‌بندی

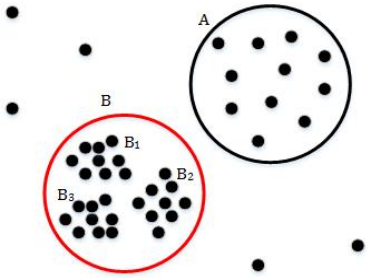
DBSCAN، الگوریتم پایهٔ روش‌های خوشه‌بندی مبتنی بر چگالی است. قبل از اینکه به بررسی مشکلات این الگوریتم بپردازیم، اجازه بدهید تا مزایای این الگوریتم را شرح دهیم. به‌طور کلی مزایای الگوریتم خوشه‌بندی DBSCAN عبارت‌اند از:

۱. الگوریتم DBSCAN توانایی کشف خوشه‌های با اشکال و اندازه‌های اختیاری را دارد. همچنین این الگوریتم می‌تواند خوشه‌هایی را که به‌طور کامل توسط یک خوشه متفاوت احاطه شده‌اند، پیدا کند.
۲. الگوریتم DBSCAN توانایی مدیریت کردن نویز را دارد.
۳. الگوریتم DBSCAN برخلاف الگوریتم K-Means نیاز ندارد که از قبل تعداد خوشه‌ها را مشخص کنیم.
۴. الگوریتم DBSCAN تنها نیاز به دو پارامتر ورودی دارد که عمدتاً به ترتیب نقاط در پایگاه داده حساس نیست (با این حال نقاطی که بر روی لبه‌های دو خوشهٔ متفاوت قرار گرفته باشند، با توجه به ترتیب نقاط ممکن است خوشه‌شان تغییر کند).

علی‌رغم مزایای گفته‌شده، این الگوریتم کمبودهایی نیز دارد که وجود این کمبودها باعث ارائه الگوریتم‌های بسیار زیادی در جهت بهبود DBSCAN شده است. در ادامهٔ این قسمت، به بررسی این کمبودها می‌پردازیم.

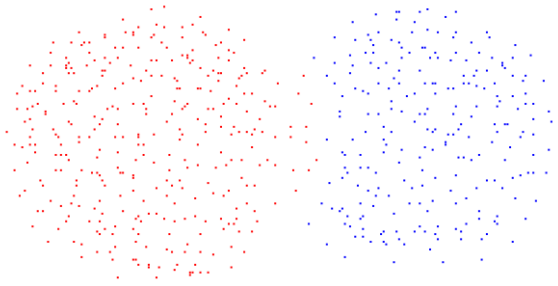
همان طور که قبلاً هم اشاره شد، الگوریتم DBSCAN نیاز به دو پارامتر ورودی Eps و Minpts دارد. خروجی این الگوریتم بسیار وابسته به انتخاب دقیق این پارامترهاست. اما تعیین دقیق این پارامترها، به‌خصوص در پایگاه داده‌های با حجم بالا، بسیار مشکل است. ازاین‌رو یکی از دغدغه‌های محققان، تعیین خودکار این پارامترها است.

یکی دیگر از معایب الگوریتم DBSCAN، عدم قابلیت تشخیص خوشه‌های با چگالی متفاوت است. این در حالی است که یکی از خصوصیات مهم مجموعه داده‌های دنیای واقعی این است که خوشه‌های موجود در این مجموعه داده‌ها، به‌دلیل وجود چگالی‌های محلی متفاوت، فقط با یک تنظیم پارامتر جهانی قابل‌تشخیص نیستند. بنابراین ما نیاز به بیش از یک چگالی محلی برای تشخیص خوشه‌ها داریم. برای نمونه، در مجموعه داده‌ای که در شکل (۵) نشان داده شده است، خوشه‌های A، B1، B2 و B3 فقط با یک تنظیم پارامتر جهانی قابل‌تشخیص نیستند. اگر پارامترها را مطابق با چگالی محلی خوشه‌های B1، B2 و B3 تنظیم کنیم، خوشهٔ A به‌عنوان نویز محسوب می‌شود. اگر پارامترها را مطابق با چگالی محلی خوشهٔ A تنظیم کنیم، خوشه‌های B1، B2 و B3 به‌اشتباه با هم ترکیب می‌شوند. بنابراین با یک تنظیم پارامتر جهانی نمی‌توان خوشه‌ها را به‌درستی تشخیص داد.



شکل (۵): خوشه‌های با چگالی متفاوت

یکی دیگر از مشکلاتی که ممکن است هنگام کار با DBSCAN با آن مواجه شویم، عدم‌تشخیص صحیح خوشه‌ها در هنگام نزدیک بودن خوشه‌هاست. همان گونه که در شکل (۶) مشاهده می‌کنید، به‌دلیل نزدیک بودن دو خوشه، الگوریتم DBSCAN نتوانسته خوشه‌ها را به‌درستی تشخیص دهد.



شکل (۶): نتایج الگوریتم DBSCAN با پارامترهای Eps=20 و Minpts=30 بر روی مجموعه داده ساختگی

الگوریتم‌های خوشه‌بندی نیاز به محاسبات هزینه‌بری دارند؛ زیرا بسیاری از این الگوریتم‌ها نیازمند رویه‌های تکرارشونده و بازگشتی دارند و همچنین اکثر داده‌های دنیای واقعی ابعاد بالایی دارند و حجم این داده‌ها نیز به‌دلیل تولید رو به رشد نقشه‌ها و اطلاعات دریافتی از ماهواره‌ها و دیگر

تجهیزات خودکار بسیار بالاست. الگوریتم DBSCAN نیز از این قاعده مستثنا نیست. قسمت زمان‌بر این الگوریتم مربوط به پرس‌وجوهای ناحیه‌ای صورت گرفته است. این الگوریتم نیاز دارد تا به‌ازای هریک از نقاط موجود در پایگاه داده یک پرس‌وجوی ناحیه‌ای انجام دهد که در مواجهه با پایگاه داده‌های با حجم و ابعاد بالا زمان انجام این عمل بسیار قابل‌توجه خواهد بود. ازاین‌رو لازم است که با استفاده از روش‌هایی پیچیدگی زمانی الگوریتم DBSCAN بهبود داده شود.

موازی‌سازی الگوریتم DBSCAN یکی از تکنیک‌های مورد استفاده برای افزایش سرعت الگوریتم DBSCAN است. رویکرد موازی‌سازی الگوریتم‌های خوشه‌بندی یک رویکرد بسیار کاربردی است و تا به امروز الگوریتم‌های خوشه‌بندی موازی متفاوتی پیاده‌سازی شده‌اند و در کاربردهای مختلف به کار گرفته شده‌اند. چندین نسخهٔ موازی از الگوریتم DBSCAN نیز ارائه شده است که در ادامه به معرفی آن‌ها خواهیم پرداخت.

برای رفع هریک از مشکلات گفته‌شده در بالا الگوریتم‌های متعددی ارائه شده است که در ادامه با این الگوریتم‌ها آشنا می‌شویم.

۶. بهبودهای ارائه‌شده

از سال ۱۹۹۶ که الگوریتم DBSCAN ارائه شده است، تاکنون الگوریتم‌های بسیار زیادی در جهت بهبود DBSCAN ارائه شده است. با این حال هنوز الگوریتم واحدی برای رفع همهٔ این مشکلات به‌طور کامل، ارائه نشده است. در این قسمت ما به معرفی و بررسی برخی از این بهبودها می‌پردازیم.

۱.۶. الگوریتم GDBSCAN

الگوریتم DBSCAN متکی بر مفهوم چگالی خوشه‌هاست و به‌منظور کشف خوشه‌های با اشکال مختلف به همراه نویز است. در [۲۳] الگوریتمی ارائه شده است که قادر است اشکال هندسی غیر از نقطه مانند چندضلعی‌های دوبعدی را نیز خوشه‌بندی کند. شبه کد این الگوریتم در شکل (۷) نشان داده شده است.

روال کلی این الگوریتم مشابه با الگوریتم DBSCAN است. اما همان طور که گفتیم این الگوریتم قابلیت خوشه‌بندی اشکال هندسی مختلف مانند چندضلعی‌ها را نیز دارد. به همین دلیل تعاریف الگوریتم DBSCAN تعمیم داده شده‌اند.

Algorithm #2: GDBSCAN Algorithm
Input: SetOfObjects, NPred, MinCard, wCard

```
1. ClusterId := nextId(NOISE);
2. FOR i FROM 1 TO SetOfObjects.size DO
3.   Object := SetOfObjects.get(i);
4.   IF Object.ClId = UNCLASSIFIED THEN
5.    IF
ExpandCluster(SetOfObjects,Object,ClusterId,
NPred,MinCard,wCard) THEN
6.      ClusterId:=nextId(ClusterId)
7.    END IF
8.   END IF
9. END FOR
10. END; // GDBSCAN
// ExpandCluster Function
ExpandCluster(SetOfObjects, Object, ClId, NPred,
MinCard, wCard)
11. IF wCard({Object}) ≤ 0 THEN
12.   SetOfPoints.changeClId(Object,
UNCLASSIFIED);
13.   RETURN False;
14. END IF
15. seeds:=SetOfObjects.neighborhood(Object,NPred)
16. IF wCard(seeds) < MinCard THEN // no core
point
17.   SetOfObjects.changeClId(Object,NOISE);
18.   RETURN False;
19. END IF
20. SetOfObjects.changeClIds(seeds,ClId);
21. seeds.delete(Object);
22. WHILE seeds ≠ Empty DO
23.   currentObject := seeds.first();
24.   result :=SetOfObjects.neighborhood
(currentObject, NPred);
25.   IF wCard(result) ≥ MinCard THEN
26.    FOR i FROM 1 TO result.size DO
27.      P := result.get(i);
28.      IF wCard({P}) > 0 AND P.ClId IN
{UNCLASSIFIED, NOISE} THEN
29.        IF P.ClId = UNCLASSIFIED THEN
30.          seeds.append(P);
31.        END IF;
32.        SetOfObjects.changeClId(P,ClId);
33.      END IF; // wCard > 0 and UNCLASSIFIED
or NOISE
34.    END FOR;
35.   END IF; // wCard ≥ MinCard
36. seeds.delete(currentObject);
37. END WHILE; // seeds ≠ Empty
38. RETURN True;
39. END; // ExpandCluster
```

شکل (۷): الگوریتم GDBSCAN

در الگوریتم DBSCAN همسایه‌های یک نقطه، نقاطی بودند که فاصله‌شان از آن نقطه کمتر از Eps بود. در اینجا دیگر از Eps استفاده نمی‌شود و به‌جای آن از NPred که یک گزارهٔ دودویی است استفاده می‌شود؛ یعنی اگر گزاره NPred(p,q) برابر با یک باشد، آنگاه دو شیء p و q همسایه هستند. این گزاره برای چندضلعی‌ها می‌تواند به این صورت تعریف شود که شیء p همسایهٔ شیء q است اگر این دو شیء با هم تداخل داشته باشند. در صورتی که wCard(p)≥MinCard باشد، شیء p یک شیء مرکزی خواهد بود.

پارامتر wCard یک اشاره‌گر به تابع wCard است که کاردینالتی وزن‌دار اشیاء را برمی‌گرداند. با توجه به خوشه‌بندی اشیائی مانند چندضلعی‌ها، تابع wCard می‌تواند مجموع نواحی اشیاء را برگرداند و پارامتر MinCard نیز حداقل ناحیه را مشخص می‌کند. بنابراین اگر مجموعه نواحی اشیاء همسایه یک شیء بزرگ‌تر یا مساوی با MinCard بود، آن شیء یک شیء مرکزی است.

الگوریتم با یک شیء اختیاری p کار را آغاز می‌کند و همهٔ اشیاء دسترسی‌پذیر چگالی از شیء p را با توجه به NPred و MinWeight (در صورتی که wCard(S)≥MinCard باشد این تابع مقدار True را برمی‌گرداند) بازیابی می‌کند. اگر p یک شیء مرکزی باشد، یک خوشه تشکیل می‌شود. در غیر این صورت همهٔ اشیاء دسترسی‌پذیر از p و خود p برچسب نویز می‌گیرند. این فرایند تا زمانی تکرار می‌شود که همهٔ اشیاء پردازش شوند.

این الگوریتم توانایی این را دارد تا اشیاء نقطه را به خوبی اشیاء بسط‌یافتهٔ مکانی بر طبق هر دو خصوصیت مکانی و غیرمکانی آن اشیاء خوشه‌بندی کند. این الگوریتم در پایگاه داده‌های بسیار بزرگ کارایی خوبی دارد. علاوه بر این، در این مقاله کاربردهایی از دنیای واقعی، مانند علوم زمین، زیست‌شناسی، نجوم و جغرافیا برای این الگوریتم ارائه شده است.

۲.۶. الگوریتم OPTICS

الگوریتم OPTICS [۹] الگوریتم DBSCAN را برای حل مسئله تغییر چگالی تطبیق داده است. شبه کد این الگوریتم در شکل (۸) نشان داده شده است.

Algorithm #3: OPTICS Algorithm
Input: SetOfObjects, Eps, Minpts, OrderedFile

```
1. OrderedFile.open();
2. FOR i FROM 1 TO SetOfObjects.size DO
3.  Object := SetOfObjects.get(i);
4.  IF NOT Object.Processed THEN
5.    ExpandCIusterOrder(SetOfObjects, Object, Eps,
MinPts, OrderedFile)
7. OrderedFile.close();
8. END; // OPTICS
//ExpandClusterOrder Function
ExpandClusterOrder(SetOfObjects, Object, Eps,
MinPts, OrderedFile);
9. neighbors := SetOfObjects.neighbors(Object, Eps);
10. ObjectProcessed := TRUE;
11. Object.reachability_distance := UNDEFINED;
12. Object.setCoreDistance(neighbors, Eps, MinPts);
13. OrderedFile.write(Object);
14. IF Object.core_distance <> UNDEFINED THEN
15.   OrderSeeds.update(neighbors, Object):
16.   WHILE NOT OrderSeeds.empty() DO
17.     currentobject := OrderSeeds.next();
18.     neighbors:=SetOfObjects.neighbors(
    currentObject , Eps);
19.     currentObject.Processed := TRUE;
20.     currentObject.setCoreDistance(neighbors, Eps,
    MinPts);
21.     OrderedFile.write(currentObject);
22.     IF
    currentObject.core_distance<>UNDEFINED
    THEN
23.       OrderSeeds.update(neighbors,
    currentobject);
24. END // ExpandClusterOrder
```

شکل (۸): الگوریتم OPTICS

این الگوریتم برای حل مسئلهٔ تغییر چگالی دو فیلد اضافی فاصلهٔ دسترسی‌پذیری (Reachability Distance) و فاصلهٔ مرکز (Core Distance) را ذخیره می‌کند. فاصلهٔ مرکز یک شیء مانند p عبارت است از فاصلهٔ شیء p از Minpts اُمین همسایه‌اش به شرطی که p یک شیء مرکزی باشد. همچنین، فاصلهٔ دسترسی‌پذیری شیء p از شیء o، کوچک‌ترین فاصله‌ای است که در آن فاصله، شیء p دسترسی‌پذیر چگالی مستق اویم از o باشد به شرطی که o یک شیء مرکزی باشد. با استفاده از این فاصله‌ها، OPTICS یک مرتب‌سازی که نشان‌دهندهٔ ساختار خوشه‌بندی است، بر روی داده‌ها می‌سازد.

در تابع ExpandClusterOrder ابتدا همسایه‌های شعاع Eps نقطهٔ انتخابی بازیابی می‌شوند و فاصلهٔ مرکز این نقطه نیز تعیین می‌شود. سپس این نقطه در فایل نوشته می‌شود. در صورتی که این نقطه یک نقطهٔ مرکزی نباشد، الگوریتم به سراغ نقطهٔ بعدی از مجموعهٔ داده می‌رود. در غیر این صورت، همهٔ اشیاء دسترسی‌پذیر چگالی مستقیم از این شیء بازیابی می‌شوند و برای بسط بیشتر در آینده به OrderSeeds اضافه می‌شوند. اشیاء موجود در این لیست بر اساس فاصلهٔ دسترسی‌پذیری‌شان از نزدیک‌ترین شیء مرکزی که از آن دسترسی‌پذیر چگالی مستقیم هستند، مرتب می‌شوند. سپس شیء با کمترین فاصلهٔ دسترسی‌پذیری از این لیست انتخاب می‌شود (مرحلهٔ ۱۶ از شکل ۸) و همسایه‌های شعاع Eps و فاصلهٔ مرکز آن تعیین می‌شوند. سپس این نقطه در فایل نوشته می‌شود. در صورتی که این نقطه یک نقطهٔ مرکزی باشد، لیست OrderSeeds بروزرسانی می‌شود تا کاندیدای دیگری برای بسط در آینده، در صورت وجود، به آن افزوده شوند. در هنگام بروزرسانی لیست OrderSeeds، ترتیب اشیاء ممکن است تغییر کند.

در این الگوریتم پارامتر Eps برای تشخیص گودی‌ها در Reachabiliy Plot که نشان‌دهندهٔ خوشه‌ها هستند، ضروری است. الگوریتم OPTICS به‌جای تولید خوشه‌های با چگالی محلی مشابه، فقط خوشه‌های با چگالی محلی بیش از یک حد آستانه را تولید می‌کند. در پایگاه داده‌های با اندازهٔ متوسط OPTICS زمان اجرا تقریباً ۱٫۶ برابر DBSCAN را دارد.

۳.۶. الگوریتم FDBSCAN

همان گونه که در بخش قبلی نیز اشاره کردیم، یکی از مشکلات الگوریتم DBSCAN بحث پیچیدگی زمانی بالای این الگوریتم در مجموعه داده‌های با حجم و بعد بالاست. الگوریتم FDBSCAN [۲۴] یکی از الگوریتم‌هایی است که باهدف بهبود سرعت DBSCAN ارائه شده است. شبه کد این الگوریتم در شکل (۹) نشان داده شده است.

Algorithm #4: FDBSCAN Algorithm
Input: SetOfObjects, Eps, MinPts

```
1. sort(SetofObjects.X);
2. ClusterId := NOISE;
3. FOR i FROM 1 TO SetofObjects.size DO
4.   Object := SetofObjects.get(i);
5.   IF Object.ClId = UNCLASSIFIED THEN
6.     seeds:=SetofObjects.RegionQuery(Object,Eps);
7.     IF seeds.size<MinPts THEN
8.       SetofObjects.ChangeClId(Object, NOISE);
9.     ELSE
10.      OldClusterId :=getfirstcoreId(seeds);
11.      IF OldClusterId =UNCLASSIFIED THEN
12.        ClusterId := nextId(ClusterId);
13.        SetofObjects.ChangeClId(seeds, ClusterId);
14.      ELSE
15.        ExpandCluster(SetofObjects, seeds,
OldClusterId, Eps, MinPts);
16.      ENDIF
17.    END IF
18.  END IF
19. END FOR
20. ReOrganize(SetofObjects);
21. END // FDBSCAN
ExpandCluster (SetofObjects, seeds, ClId,
Eps,MinPts)
22. WHILE seeds <> Empty
23.   currentO := seeds.first();
24.   IF SetofObjects.IsCoreObject(currentO, Eps, MinPts)
25.     SetofObjects.MergeCluster(currentO, ClId);
26.   ELSE
27.     SetofObjects.changeClId(currentO,ClId);
28.   END IF
29.   seeds.delete(currentO);
30. END WHILE
31. END // ExpandCluster
```

شکل (۹): الگوریتم FDBSCAN

با توجه به اینکه بسیاری از پرس‌وجوهای ناحیه‌ای برای یافتن همسایه‌های اشیاء قابل چشم‌پوشی هستند، ما باید تعدادی شیء را به‌جای تمامی اشیاء انتخاب کنیم که پرس‌وجوی ناحیه‌ای فقط بر روی آن اشیاء انجام شود. الگوریتم FDBSCAN بر پایه دو قضیه زیر حاصل شده است.

قضیهٔ ۱: فرض کنید p و q دو شیء مرکزی باشند، در صورتی که یک شیء مرکزی مثل o وجود داشته باشد که بین همسایه‌های p و q مشترک باشد، ما می‌توانیم دو خوشهٔ مربوط به p و q را ادغام کنیم.

قضیهٔ ۲: در صورتی که p یک شیء مرکزی باشد، همهٔ اشیاء همسایهٔ p متعلق به یک خوشه هستند.

ابتدا اشیاء بر اساس مختصات xشان مرتب می‌شوند. سپس شیء p با کمترین اندیس را انتخاب کرده و همه همسایه‌های شعاع Eps آن را بازیابی می‌کنیم. اگر شیء p یک شیء مرکزی نباشد، برچسب نویز می‌گیرد. در غیر این صورت، از میان اشیاء همسایه p، اولین شیئی را که برچسب خوشه دارد (مانند شیء q) را پیدا می‌کنیم (مرحله ۱۰ از شکل (۹)). اگر چنین شیئی وجود نداشته باشد، یعنی هیچ‌کدام از همسایه‌های شیء p برچسب خوشه نداشته باشند، آنگاه یک خوشهٔ جدید ایجاد می‌شود و همه همسایه‌های p بعلاوه خود p برچسب خوشهٔ جدید را می‌گیرند. در غیر این صورت، p و همه همسایه‌های آن برچسب خوشه مربوط به شیء q را می‌گیرند. به‌طور همزمان، اگر هرکدام از همسایه‌های دیگر شیء p به‌غیراز q، شیء مرکزی باشند، خوشه مربوطه‌شان با خوشه مربوط به شیء q ادغام می‌شود. این فرایند تا زمانی ادامه می‌یابد که همه اشیاء پردازش شوند.

پیچیدگی زمان اجرا و زمان محاسباتی الگوریتم FDBSCAN نسبت به الگوریتم پایه DBSCAN بهتر است. به‌طور کلی کارایی این الگوریتم نسبت به الگوریتم DBSCAN بهتر است، با این حال در این الگوریتم اشیاء بیشتری نسبت به الگوریتم DBSCAN از دست می‌روند.

۴.۶. الگوریتم DBRS

در [۲۵] الگوریتم دیگری به نام DBRS ارائه شده است که هدف آن کاهش زمان اجرا برای مجموعه داده‌های با چگالی متفاوت است. DBRS معتقد است که با تعداد کمی نقطهٔ مرکزی (نقاط اسکلت) و همسایه‌هایشان می‌توان خوشه‌ها را تشخیص داد. با توجه به اینکه یافتن نقاط اسکلتی یک مسئله NP-Complete است، ازاین‌رو نقاط نمونه به‌صورت تصادفی انتخاب می‌شوند. شبه کد این الگوریتم در شکل (۱۰) نشان داده شده است.

الگوریتم با یک نقطهٔ اختیاری q شروع می‌کند و همسایه‌های تطبیقی آن نقطه را بازیابی می‌کند. همسایه‌های تطبیقی نقطهٔ q شامل نقاطی مانند p می‌شود، به‌گونه‌ای که فاصلهٔ p از q کمتر از Eps باشد و همچنین خصوصیت غیرمکانی نقاط p و q مشابه باشد. در صورتی که تعداد همسایه‌های تطبیقی q کمتر از Minpts باشد یا مقدار خلوص لیست همسایه‌های تطبیقی نقطهٔ q (qseed) کمتر از MinPur باشد، آن نقطه، نویز یا نقطهٔ حاشیه‌ای در نظر گرفته می‌شود.

Algorithm #5: DBRS Algorithm
Input: D, Eps, MinPts, MinPur

```
1. ClusterList = Empty;
2. while (!D.isClassified( ))
3.   Select one unclassified point q from D;
4.   qseeds = D.matchingNeighbors(q, Eps);
5.   if ((|qseeds| < MinPts) or (qseeds.pur < MinPur))
6.     q.clusterID = -1;  /*q is noise or a border point */
7.   else
8.     isFirstMerge = True;
9.     Ci = ClusterList.firstCluster;
/* compare qseeds to all existing clusters */
10.    while (Ci != Empty)
11.      if ( hasIntersection(qseeds, Ci) )
12.        if (isFirstMerge)
13.          newCi = Ci.merge(qseeds);
14.          IsFirstMerge = False;
15.        else
16.          newCi = newCi.merge(Ci);
17.          ClusterList.deleteCluster(C);
18.        End If
19.      End If
20.      Ci = ClusterList.nextCluster;
21.    End While // while (Ci!=Empty)
/*No intersection with any existing cluster */
22.    if (isFirstMerge)
23.      Create a new cluster Cj from qseeds;
24.      ClusterList = ClusterList.addCluster(Cj);
25.    End If // if (isFirstMerge)
26.  End If // Else
27. End While // while (!D.isClassified)
28.End // DBRS
```

شکل (۱۰): الگوریتم DBRS

مقدار خلوص همسایه‌های تطبیقی یک نقطه از تقسیم تعداد همسایه‌های تطبیقی آن نقطه بر تعداد همسایه‌های شعاع Eps آن نقطه به دست می‌آید. از پارامتر MinPur برای کنترل خلوص (سازگاری) همسایه‌های یک نقطه استفاده شده است. در صورتی که نقطهٔ انتخابی یک نقطهٔ مرکزی باشد، الگوریتم بررسی می‌کند که آیا همسایه‌های نقطهٔ مرکزی با هیچ‌یک از خوشه‌های موجود اشتراک دارند یا خیر. در صورتی که

qseed با خوشه‌های موجود تداخل داشته باشد، با آن خوشه‌ها ادغام می‌شود و در غیر این صورت نقاط موجود در qseed خوشهٔ جدیدی را تشکیل می‌دهند.

یکی از ویژگی‌های الگوریتم DBRS قابلیت پشتیبانی از ویژگی‌های غیرمکانی اشیاء است. DBRS نسبت به DBSCAN مقیاس‌پذیرتر است و زمان اجرای کمتری دارد. این الگوریتم برای خوشه‌های با چگالی بالا کارایی بهتری دارد؛ چراکه با افزایش مقدار Eps نیاز به پرس‌وجوی کمتری دارد. تنها مشکلی که ممکن است در این الگوریتم رخ دهد، عدم توانایی در ادغام کردن خوشه‌های کوچک است.

۵.۶. الگوریتم VDBSCAN

الگوریتم VDBSCAN [۲۶] به‌منظور رفع مشکل DBSCAN در تجزیه‌وتحلیل خوشه‌های با چگالی متفاوت ارائه شده است. شبه کد این الگوریتم در شکل (۱۱) نشان داده شده است.

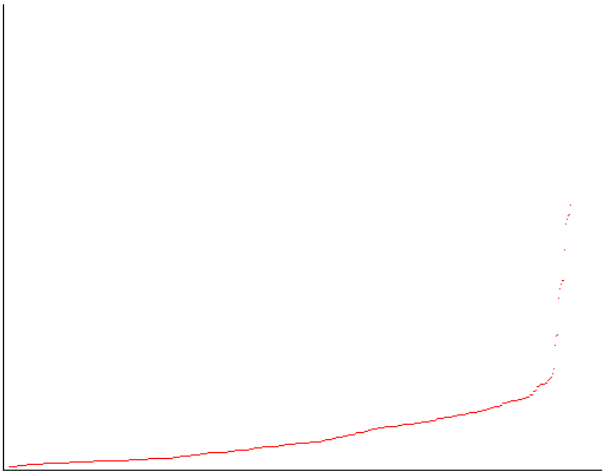
Algorithm #6: VDBSCAN Algorithm
Input: D, k

1. Partition k-dist plot;
2. Give thresholds of parameters Eps_i (i=1 ,2,…,n);
3. For each Eps_i (i=1,2,…,n)
4. Eps=Eps_i;
5. Adopt DBSCAN algorithm for points that are not marked;
6. Mark points as Ci-t;
7. Display all the masked points as corresponding clusters

شکل (۱۱): الگوریتم VDBSCAN

ایدهٔ این الگوریتم به این صورت است که قبل از اعمال الگوریتم DBSCAN با استفاده از مفهوم k-dit plot چگالی‌های مختلف را شناسایی کرده و برای هر چگالی یک مقدار Eps متناسب را برگزینند. بعد از تعیین مقادیر مختلف Eps، الگوریتم DBSCAN را به تعداد چگالی‌های به‌دست‌آمده با استفاده از مقادیر مختلف Eps به‌دست‌آمده بر روی مجموعه داده اعمال می‌کند. منحنی k-dist plot از مرتب‌سازی نقاط مجموعه داده بر اساس فاصلهٔ هر نقطه از kامین نزدیک‌ترین همسایه‌اش ساخته می‌شود. نویسنده معتقد است که هر تغییر شدید در این منحنی تعیین‌کنندهٔ یک چگالی است.

با پیاده‌سازی الگوریتم VDBSCAN و انجام تست بر روی مجموعه داده‌های مختلف، فهمیدیم که این الگوریتم فقط برای برخی مجموعه داده‌های خاص، که دارای منحنی ملایم در k-dist plot متناظرشان نیستند، صحیح کار می‌کند و بر روی بسیاری از مجموعه داده‌ها قادر به تشخیص صحیح پارامترها نیست. در واقع با توجه به اینکه الگوریتم VDBSCAN با توجه به تغییرات شدید مشاهده‌شده در منحنی k-dist plot چگالی‌های مختلف را تشخیص می‌دهد، در مجموعه داده‌هایی که منحنی k-dist plot آن‌ها همانند شکل (۱۲) دارای شیب ملایم باشد، تعیین چگالی‌های متفاوت مشکل است.



شکل (۱۲): منحنی k-dist Plot مربوط به مجموعه داده can473 با فرض k=8

الگوریتم VDBSCAN توانایی کشف اشکال اختیاری از خوشه‌ها را دارد و در مقابل نویز قوی است. این الگوریتم پیچیدگی زمانی مشابه DBSCAN دارد ولی برخلاف DBSCAN قابلیت تشخیص خوشه‌های با چگالی متفاوت را دارد و همچنین پارامتر Eps را نیز به‌صورت خودکار تعیین می‌کند. نیاز به پارامتر ورودی k یکی از نقاط ضعف این الگوریتم است به گونه‌ای که عدم انتخاب صحیح آن باعث تنزل دقت نتایج می‌شود. همچنین، همان گونه که قبلاً نیز اشاره کردیم، این الگوریتم تنها برای برخی مجموعه داده‌های خاص، که دارای منحنی ملایم در k-dist Plot متناظرشان نیستند، صحیح کار می‌کند و بر روی بسیاری از مجموعه داده‌ها قادر به تشخیص صحیح پارامترها نیست.

۱.۵.۶. کارهای آتی

الگـوریتم VDBSCAN نیـاز بـه یـک پـارامتر ورودی k دارد. خروجی این الگوریتم وابسته بـه انتخـاب دقیـق پـارامتر k دارد و تعیین این پارامتر توسط کاربر به‌خصـوص در مجموعـه داده‌هـای بزرگ مشکل است. ازاین‌رو، ارائهٔ یک روش کارآمد برای انتخـاب خودکار این پارامتر یکی از زمینه‌های تحقیقاتی پیش روست.

در [۲۷] بهبودی از الگوریتم DBSCAN ارائه شده که هـدف آن کـاهش پارامترهـای ورودی و در نتیجـه کـاهش خطاهـای ایجادشده به‌دلیل دخالت کاربر است. روال کار این الگوریتم شـبیه بـه روال کـار الگـوریتم VDBSCAN اسـت. امـا ایـن الگـوریتم برخلاف الگـوریتم VDBSCAN کـه فقـط kاُمـین نزدیـک‌تـرین همسایه را در محاسبهٔ فاصله در نظر می‌گیـرد، میـانگین فاصـله از همهٔ k نزدیک‌ترین همسـایه‌هـا را لحـاظ مـی‌کنـد. ایـن کـار بـه تشخیص نویز کمک می‌کند و باعث می‌شود تا تشخیص خودکـار حد آستانهٔ چگالی‌ها ساده‌تـر شـود. ایـن الگـوریتم نتـایج بهتـری نسبت به الگوریتم DBSCAN تولید می‌کند منتها هنوز نیاز به یـک پارامتر k دارد که باید توسط کاربر تعیین شـود. همچنـین در [۲۸] یکـی دیگـر از بهبودهـای الگـوریتم DBSCAN تحـت عنـوان KDDClus ارائه شده است که رول کار این الگوریتم نیـز هماننـد روال کار الگوریتم ارائه‌شده در [۲۷] است بـا ایـن تفـاوت کـه در این الگـوریتم از سـاختار داده KD-Tree بـرای پـردازش کـارای داده‌های با ابعاد بالا استفاده شده است. در واقع استفاده از سـاختار دادهٔ KD-Tree محاسبهٔ کارای Kاُمین نزدیک‌ترین همسـایه‌هـا را، به‌خصوص برای مجموعه داده‌های بزرگ ممکن می‌سازد.

۶.۶. الگوریتم ST-DBSCAN

ST-DBSCAN [۲۹] یکـی دیگـر از توسـعه‌هـای الگـوریتم DBSCAN است که برخلاف الگوریتم DBSCAN قابلیت کشف خوشه‌ها مطابق با مقادیر مکانی، غیرمکانی و زمـانی اشـیاء را دارد. این الگوریتم از سه جهت نسبت به الگـوریتم DBSCAN تفـاوت دارد. اول اینکه ایـن الگـوریتم بـرخلاف الگـوریتم DBSCAN و دیگر الگوریتم‌های مبتنی بر چگالی موجود، قابلیـت خوشـه‌بنـدی داده‌های مکانی‌زمانی را بر طبق خصوصیات مکـانی، غیرمکـانی و زمانی اشیاء را دارد. دوم اینکه این الگوریتم بـرخلاف DBSCAN در مواقعی که خوشه‌هـای بـا چگـالی متفـاوت در مجموعـه داده وجود داشته باشند نیز قابلیت تشخیص نویز را دارد. همچنین اگـر مقادیر غیرمکـانی اشـیاء همسـایه تفـاوت انـدکی داشـته باشـند و خوشـه‌هـا مجـاور یکـدیگر باشـند، مقـادیر اشـیاء حاشـیه‌ای در یک‌طرف خوشه ممکـن اسـت بسـیار متفـاوت بـا مقـادیر اشـیاء حاشیه‌ای در طرف مقابـل باشـند. الگـوریتم ST-DBSCAN ایـن مشکل را با مقایسه مقدار میـانگین یـک خوشـه بـا مقـادیر اشـیاء جدید اضافه‌شده، حل می‌کند. شـبه کـد ایـن الگـوریتم در شـکل (۱۳) نشان داده شده است.

روال کلـی ST-DBSCAN مشـابه الگـوریتم DBSCAN است با این تفاوت که در زمان انجـام پـرس‌وجـوی ناحیـه‌ای، جسـتجوی همسـایه‌هـا بـر اسـاس دو مقـدار Eps1 (ویژگـی مکانی) و Eps2 (ویژگی غیرمکانی مانند دما) انجام می‌شود.

Algorithm #7: ST-DBSCAN Algorithm
Input: SetOfPoints, Eps1, Eps2, MinPts, $\Delta\varepsilon$

```
1. Cluster_Lable=0
2. For i=1 to n
3.   If o_i is not in a cluster Then
4.     X=Retrieve_Neighbors(o_i,Eps1,Eps2)
5.   If |X| < Minpts Then
6.     Mark o_i as noise
7.   Else
8.     Cluster_Lable=Cluster_Lable+1
9.    For J=1 to |X|
10.    Mark all objects in X with current Cluster_Lable
11.  End For
12.  Push(all objects in X)
13.  While not IsEmpty()
13.    CurrentObj=Pop()
14.    Y=Retrieve_Neighbors(CurrentObj,Eps1,Eps2)
15.    if |Y| >= Minpts Then
16.      ForAll objects o in Y
17.        If (o is not marked as noise or it is not in a
cluster) and |Cluster_Avg() – o.Value| <= Δε Then
18.          Mark o with current Cluster_Lable
19.          Push (o)
20.        End If
21.      End For
22.    End If
22.   End While
23.  End If
24.  End If
25. End For
26. End // ST-DBSCAN
```

شکل (۱۳): الگوریتم ST-DBSCAN

کار با انتخاب یک نقطهٔ اختیاری شروع می‌شود. در صورتی که نقطهٔ انتخاب‌شده یک نقطهٔ مرکزی باشد، یک خوشهٔ جدید ساخته می‌شود و همهٔ اشیاء دسترسی‌پذیر چگالی مستقیم از این نقطهٔ مرکزی برچسب خوشهٔ جدید را می‌گیرند و سپس به یک پشته افزوده می‌شوند. سپس الگوریتم اشیاء دسترسی‌پذیر چگالی از شیء مرکزی انتخابی را با استفاده از پشته بازیابی می‌کند. این اشیاء در صورتی که برچسب خوشه یا نویز نداشته باشند و تفاوت بین مقدار این اشیاء و میانگین مقدار خوشه کمتر از پارامتر $\Delta\varepsilon$ باشد، به خوشه افزوده می‌شوند. در اینجا از پارامتر $\Delta\varepsilon$ برای اجتناب از کشف خوشه‌های ترکیب‌شده استفاده می‌شود. در صورتی که دو خوشه نزدیک هم باشند، ممکن است نقطه‌ای مثل p بتواند به هر دو خوشه تعلق بگیرد که در این صورت p به خوشه‌ای که زودتر آن را کشف کرده است تعلق می‌گیرد.

پیچیدگی زمانی این الگوریتم همانند DBSCAN و از مرتبه O(nlogn) است. این الگوریتم قابلیت کار بر روی داده‌های زمانی‌مکانی را دارد. دادهٔ زمانی‌مکانی یعنی داده‌هایی که به‌عنوان برش‌های زمانی از مجموعه دادهٔ مکانی ذخیره شده‌اند. کشف دانش در داده‌های زمانی‌مکانی به‌مراتب مشکل‌تر از داده‌های غیرمکانی و داده‌های زمانی است. الگوریتم ST-DBSCAN کاربردهای متعددی دارد که از جمله این کاربردها می‌توان به سیستم اطلاعات جغرافیا، تصاویر پزشکی و پیش‌بینی وضع هوا اشاره کرد. یکی از نقاط ضعف این الگوریتم عدم توانایی کشف خوشه‌های با چگالی متفاوت است. همچنین پارامترهای ورودی این الگوریتم به‌صورت خودکار تولید نمی‌شوند.

۱.۶.۶. کارهای آتی

الگوریتم ST-DBSCAN قابلیت تشخیص خوشه‌های با چگالی متفاوت را ندارد، ازاین‌رو بهبود این الگوریتم به‌منظور تشخیص خوشه‌های با چگالی متفاوت یکی از کارهای آتی پیش روی محققان است. پارامترهای ورودی این الگوریتم نیز همانند پارامترهای DBSCAN قابلیت این را دارند که به‌صورت خودکار تعیین شوند. همچنین کارایی این الگوریتم نیز می‌تواند بهبود داده شود.

۷.۶. الگوریتم LDBSCAN

الگوریتم LDBSCAN [۳۰] یکی دیگر از بهبودهای الگوریتم DBSCAN است که از مفهوم فاکتور دورافتادگی محلی [۳۱] (LOF) و چگالی دسترسی‌پذیری محلی (LRD)، برای تشخیص خوشه‌های با چگالی متفاوت استفاده می‌کند. شبه کد این الگوریتم در شکل (۱۴) نشان داده شده است.

Algorithm #8: LDBSCAN Algorithm
Input: SetOfPoints, LOFUB, pct, MinPts

```
1. InitSet (SetOfPoints);=calculate LRD and LOF of
each Points.
2. ClusterID :=0;
3. For i From 1 TO SetOfPoints.size DO
4.    Point := SetOfPoints.get(i);
5.    IF Point.ClId = UNCLASSIFIED Then
6.      IF LOF(Point) ≤LOFUB Then
7.        ClusterID := ClusterID + 1;
8.        ExpandCluster(SetOfPoints, Point,
          ClusterID, pct, MinPts);
9.      Else
10.       SetOfPoint.changeClId(Point,NOISE);
11.     End If
12    End If
13. End For
14. End //LDBSCAN
// EcpandCluster Function
ExpandCluster(SetOfPoints, Point, ClusterID, pct,
MinPts)
15. SetOfPoint.changeClId(Point,ClusterID);
16. FOR i FROM 1 TO MinPts DO
17.    currentP := Point.Neighbor(i);
18.    IF currentP.ClId IN {UNCLASSIFIED,NOISE}
and DirectReachability(currentP,Point) THEN
19.      TempVector.add(currentP);
20.      SetOfPoint.changeClId(currentP,ClusterID);
21.    END IF
22. END FOR
23. WHILE TempVector <> Empty DO
24.    Point := TempVector.firstElement();
25.    TempVector.remove(Point);
26.    FOR i FROM 1 TO MinPts DO
27.      currentP := Point.Neighbor(i);
28.      IF currentP.ClId IN{UNCLASSIFIED,NOISE}
and DirectReachability (currentP,Point) THEN
29.        TempVector.add(currentP);
30.        SetOfPoint.changeClId(currentP,ClusterID);
31.      END IF
32.    END FOR
33. END WHILE
34. END //ExpandCluster
```

شکل (۱۴): الگوریتم LDBSCAN

در این الگوریتم از فاکتور دورافتادگی محلی برای تشخیص نویز استفاده شده است، به‌گونه‌ای که اگر فاکتور دورافتادگی محلی یک نقطه کمتر از یک حد آستانه باشد، آن نقطه یک نقطهٔ مرکزی است و در غیر این صورت نویز محسوب می‌شود (مرحلهٔ ۶ از شکل ۱۴). در تابع ExpandCluster یک نقطه در صورتی بسط داده می‌شود که چگالی دسترسی‌پذیری محلی آن نقطه نزدیک به چگالی دسترسی‌پذیری محلی نقطهٔ مرکزی خوشهٔ متعلق به آن باشد. در غیر این صورت آن نقطه به‌طور ساده به خوشه افزوده می‌شود و بسط داده نمی‌شود.

علاوه بر قابلیت تشخیص خوشه‌های با چگالی متفاوت، در این الگوریتم انتخاب پارامترهای مناسب نسبت به الگوریتم DBSCAN ساده‌تر است. با این حال، این الگوریتم نیاز به چهار پارامتر ورودی دارد. هرچند که انتخاب مقادیر مناسب برای این پارامترها چندان سخت نیست، برای پایگاه داده‌های بسیار حجیم، تعداد زیاد پارامترهای ورودی ممکن است مشکل‌ساز شود

۸.۶. الگوریتم DDSC

الگوریتم DDSC [۳۲] یکی دیگر از الگوریتم‌هایی است که هدف آن یافتن خوشه‌های با چگالی متفاوت است. شبه کد این الگوریتم در شکل (۱۵) نشان داده شده است.

Algorithm #9: DDSC Algorithm
Input: SetOfPoints, Eps, MinPts, α

1. Find the neighbourhood, $N_\epsilon(p)$;
2. Set the density value, $w_p = |N_\epsilon(p)|$;
3. If p is a homogeneous core, perform steps 4-7;
4. Find the list L_p of unlabeled objects in $N_\epsilon(p)$: $L_p = \{q \mid q \in N_\epsilon(p), c_p = -1\}$;
5. Arrange the objects in Lp in ascending order of their distance to p giving the sorted list L_p with size $t = |L_p|$, that is : $L_p = \{q_i \mid q_i \in L_p, i \in 1 \ldots t, q_0 = p, dist(q_{i-1}, p) \leq dist(q_i, p)\}$;
6. Append L_p in seed list S;
7. Mark all unlabeled and noise objects in $N_\epsilon(p)$ with present cluster label c: $\forall\, q\{q \in N_\epsilon(p), c_p \leq 0\} : c_p = c$;

شکل (۱۵): الگوریتم DDSC

در این الگوریتم از مفهوم شیء مرکزی هم‌جنس استفاده شده است. شیء مرکزی هم‌جنس به شیئی گفته می‌شود که اولاً یک شیء مرکزی باشد و ثانیاً اختلاف چگالی این شیء با همسایه‌هایش در حد α باشد. الگوریتم با انتخاب یک شیء مرکزی هم‌جنس کار خودش را آغاز می‌کند و تا زمانی خوشه را بسط می‌دهد که به یک شیء مرکزی غیر هم‌جنس، که نشان‌دهندهٔ تغییر وسیع در چگالی است، برسد. در ضمن نقاط همسایهٔ نقطهٔ مرکزی هم‌جنس به ترتیب فاصله‌شان از همان نقطهٔ مرکزی هم‌جنس، برای بسط بیشتر مورد بررسی قرار می‌گیرند.

یکی از مزایای الگوریتم خوشه‌بندی DDSC کاهش وابستگی به پارامتر Eps است. همچنین، پیچیدگی زمانی این الگوریتم همانند الگوریتم DBSCAN و از مرتبهٔ O(nlogn) است.

۹.۶. الگوریتم GF-DBSCAN

الگوریتم GF-DBSCAN [۳۳] یکی دیگر از الگوریتم‌هایی است که در سال ۲۰۰۹ و با هدف بهبود زمان الگوریتم DBSCAN ارائه شد. به‌منظور رسیدن به این هدف، این الگوریتم از مفهوم Grid-Based استفاده می‌کند. به این صورت که الگوریتم ابتدا فضای داده را به سلول‌های شبکه‌ای با عرض Eps تقسیم می‌کند. در این حالت برای یافتن همسایه‌های یک نقطه، دیگر نیاز نیست که همهٔ نقاط بررسی شوند، بلکه تنها همسایه‌های سلول شبکه‌ای آن نقطه مورد بررسی قرار می‌گیرند. شبه کد این الگوریتم در شکل (۱۶) نشان داده شده است.

Algorithm #10: GF-DBSCAN Algorithm
Input: D, Minpts, Eps

1. Select datasets and initialization parameters.
2. Partitioning the data into several cells according to the parameter Eps.
3. For Each Point p in D
4. Find neighborhoods Cells, $NC_\epsilon(p)$
5. If $|NC_\epsilon(p)| <$ Minpts Then
6. Mark p as noise
7. Else
8. P and $NC_\epsilon(p)$ regarded as a new cluster C
9. If the points in a new cluster do not involve any other cluster points Then
10. Create a new independent cluster
11. Else
12. If intersection point is a Core Point Then
13. Merge the sub-clusters into a single cluster
14. If not all points are defined yet, then
15. return to Step3 until all points are defined, and stop search.

شکل (۱۶): الگوریتم GF-DBSCAN

الگوریتم ابتدا مجموعه داده را به سلول‌های شبکه‌ای با اندازهٔ Eps تقسیم می‌کند. سپس با خواندن یک نقطه و پیدا کردن سلول‌های همسایهٔ این نقطه، در صورتی که تعداد نقاط موجود در این سلول‌ها کمتر از Minpts باشد، آن نقطه به‌عنوان نویز برچسب می‌خورد و در غیر این صورت نقطهٔ انتخابی به همراه همسایه‌های آن به‌عنوان یک خوشهٔ جدید در نظر گرفته می‌شوند. حال نوبت آن است که الگوریتم تصمیم بگیرد که آیا یک خوشهٔ مستقل را ایجاد کند یا چند زیرخوشه را ادغام کند. اگر نقاط موجود در خوشهٔ جدید شامل هیچ نقطه‌ای از خوشه‌های دیگر نبود، یک خوشهٔ مستقل ایجاد می‌شود. در غیر این صورت، سلول‌های همسایهٔ نقطهٔ مشترک بین خوشهٔ جدید و خوشه‌های موجود بازیابی می‌شوند. در صورتی که تعداد نقاط موجود در این سلول‌های همسایه بیشتر از Minpts باشند، یعنی نقطهٔ اشتراک یک نقطهٔ مرکزی باشد، زیرخوشه‌های شامل این نقطهٔ اشتراک به داخل یک خوشه ادغام می‌شوند. این فرایند تا زمانی ادامه می‌یابد که همه نقاط پردازش شوند.

شایان ذکر است که زمان الگوریتم GF-DBSCAN یک‌صدم زمان الگوریتم FDBSCAN است. علاوه بر این، نرخ صحیح بودن خوشه‌بندی (CCR) و نرخ فیلتر کردن نویز (NFR) در این الگوریتم مشابه با الگوریتم FDBSCAN است. بنابراین می‌توانیم بگوییم که الگوریتم GF-DBSCAN یک الگوریتم خوشه‌بندی است که خوشه‌های پایدار را تولید می‌کند.

۱۰.۶. الگوریتم ODBSCAN

در [۳۴] الگوریتمی ارائه شده است که هدف آن بهبود سرعت الگوریتم DBSCAN است. این بهبود در سرعت، از طریق کاهش تعداد فراخوانی‌های پرس‌وجوهای ناحیه‌ای و همچنین بهبود سرعت برخی از فراخوانی‌های پرس‌وجوی ناحیه‌ای انجام شده است.

کارایی این الگوریتم نسبت به الگوریتم‌های موجود بهتر است و پیچیدگی زمانی آن نیز نسبت به DBSCAN و FDBSCAN کمتر است. همچنین اشیاء ازدست‌رفته در این الگوریتم نسبت به FDBSCAN کمتر است چراکه اشیاء حاشیه‌ای نیز در این الگوریتم لحاظ می‌شوند. ازاین‌رو الگوریتم ODBSCAN دقت بالاتری نسبت به الگوریتم FDBSCAN دارد.

۱۱.۶. الگوریتم DVBSCAN

الگوریتم DVBSCAN [۳۵] یکی دیگر از الگوریتم‌هایی است که به‌منظور رفع مشکل تغییرات چگالی الگوریتم DBSCAN ارائه شده است. این الگوریتم از مفهوم واریانس چگالی خوشه (CDV) و شاخص شباهت خوشه (CSI)، به‌منظور جلوگیری از بسط خوشه از ناحیهٔ متراکم به ناحیهٔ متراکم‌تر و برعکس استفاده می‌کند. شبه کد این الگوریتم در شکل (۱۷) نشان داده شده است.

Algorithm #11: DVBSCAN Algorithm
Input: D, Minpts, Eps, α, γ

```
1. Initially all objects are unclassified
2. For each unclassified object x ∈ D
3.   If Core(x) Then
4.     Generate new ClusterID & Assign the clusterID to x
5.     Insert x into the Queue
6.     While Queue ≠ Empty
7.       Extract front object y from the Queue
8.       Calculate S={o ∈ D | dist (y, o) ≤ Eps}
9.       For each object o ∈ S
10.        If o is unclassified and DCCO(o) Then
11.          insert o into Queue
12.        If o is unclassified or noise Then
13.          assign the clusterID to o
14.      End For
15.    End while
16.  Else x is noise
17. End for
18. End // DVBSCAN
```

شکل (۱۷): الگوریتم DVBSCAN

الگوریتم با انتخاب یک نقطهٔ مرکزی شروع به شکل‌دهی خوشه‌ها می‌کند. سپس همهٔ نقاطی را که در همسایگی Eps نقطهٔ مرکزی انتخابی باشند، به یک صف وارد می‌کند. این نقاط در صورتی اجازهٔ بسط پیدا می‌کنند که واریانس چگالی خوشهٔ آن‌ها کمتر یا مساوی از حد آستانهٔ α باشد و همچنین شاخص شباهت خوشه، یعنی اختلاف بین حداقل و حداکثر شیء قرار گرفته در خوشه، نیز کمتر از حد آستانهٔ γ باشد

(این دو شرط در تابع DCCO بررسی می‌شوند). در غیر این صورت، نقطه به‌طور ساده به خوشه افزوده می‌شود و دیگر بسط داده نمی‌شود. این الگوریتم علاوه بر دو پارامتر استفاده‌شده در الگوریتم DBSCAN نیاز به تعیین دو پارامتر α و γ دارد که به‌منظور محدود کردن مقدار تغییر چگالی محلی اجازه داده‌شده در داخل خوشه‌ها استفاده شوند.

الگوریتم DVBSCAN قابلیت تشخیص خوشه‌های با اندازه، اشکال و چگالی متفاوت را دارد و در مقابل نویز نیز مقاوم است. با این حال، این الگوریتم نیاز به تعیین چهار پارامتر ورودی دارد که تعیین چهار پارامتر به‌مراتب سخت‌تر از تعیین دو پارامتر نسبت به الگوریتم DBSCAN است. این در حالی است که نتایج این الگوریتم بسیار وابسته به تعیین دقیق این پارامترهاست. برای مثال اگر مقدار پارامتر α کوچک باشد، الگوریتم تعداد زیادی از خوشه‌های غیر مهم کوچک را تولید می‌کند، از طرفی اگر مقدار این پارامتر را بزرگ بگیریم، الگوریتم تعدادی از خوشه‌های باکیفیت خوب را به داخل یک خوشه ادغام می‌کند.

۱.۱۱.۶. کارهای آتی

پیچیدگی زمانی این الگوریتم با توجه به روش مورد استفاده در آن بالاست و باید کاهش پیدا کند. همچنین برای اینکه به نتایج خوشه‌بندی مناسب برسیم نیاز داریم تا پارامترهای ورودی این الگوریتم به‌خصوص α و γ را به‌صورت خودکار تعیین کنیم.

۱۲.۶. الگوریتم IVDBSCAN

الگوریتم VDBSCAN قابلیت تشخیص خوشه‌های با چگالی متفاوت را دارد و در مقابل نویز نیز مقاوم است. اما این الگوریتم تا حد زیادی به پارامترهای Eps و Minpts بستگی دارد و نیاز دارد که این دو پارامتر با دقت بسیار بالایی انتخاب شوند. در [۳۶] روشی برای انتخاب خودکار این دو پارامتر ارائه شده است. الگوریتم پیشنهادشده دارای دو فاز و به‌طور کلی، شش مرحله زیر است:

۱. پارتیشن‌بندی مجموعه داده با استفاده از فاصلهٔ اقلیدسی

۲. محاسبهٔ تفاوت بین حداقل و حداکثر نقطه جهت به دست آوردن مقدار Minpts

۳. محاسبهٔ Eps برای هر پارتیشن

۴. تطبیق DBSCAN برای هر Eps

۵. برچسب‌گذاری نقاط

۶. نمایش نقاط برچسب‌گذاری‌شده مطابق با خوشه‌های متناظرشان

این الگوریتم علاوه بر انتخاب خودکار پارامترهای ورودی Eps و Minpts، از دقت و کارایی بهتری نسبت به الگوریتم VDBSCAN برخوردار است.

۱۳.۶. الگوریتم MDDBSCAN

در [۳۷] یک بهبود از DBSCAN ارائه شده است که قابلیت تشخیص خوشه‌های متصل‌به‌هم با چگالی متفاوت را دارد. شبه کد این الگوریتم در شکل (۱۸) نشان داده شده است.

Algorithm #12: MDDBSCAN Algorithm
Input: D,k,var

1. Calculate the Euclidean distance between each two points of the dataset.
2. For each point **p**, find the average distance between it and its kth nearest neighbors:

$$DST_p = \sum_{q \in N_p} \frac{dist(p,q)}{k}$$

Where Np is the group of kth nearest neighbors of p.
3. Insert all points in the queue list.
4. Starting from the most dense point in the queue which have the smallest DST value and do the following:
   (a) Assign the point **p** to new cluster (Ci).
   (b) Remove **p** from the queue list.
   (c) The initial average distances of the cluster will be:
   $AVGDST_{Ci} = DSTp$
   (d) Call: gather(Ci, p)
5. gather (Ci, p): For each point **q** in the nearest kth-neighbors of **p** do the following:
(a) if (**q** in queue list) **and** ($DSTq \le var*AVGDST_{Ci}$) then:
   (i) Add q to Ci members
   (ii) Remove q from the queue list.
   (iii) Calculate $AVGDST_{Ci}$ from the equation:

$$AVGDST_i = \frac{AVGDST_i * (N_i - 1) + DST_q}{N_i}$$

   (iv) Call: gather(Ci, q)
   (v) End if

شکل (۱۸): الگوریتم MDDBSCAN

الگوریتم در ابتدا چگالی هر نقطه (DST_P) را محاسبه می‌کند. سپس متراکم‌ترین نقطه را به‌عنوان شیء مرکزی در نظر می‌گیرد و شروع به بسط خوشه به‌وسیلهٔ نقاط همسایه با چگالی مشابه می‌کند. در تابع gather از بین نقاطی که جز k نزدیک‌ترین همسایهٔ شیء مرکزی هستند، فقط نقاطی که چگالی‌شان کمتر از میانگین چگالی خوشه (var * AVGDST) باشد به خوشه افزوده می‌شوند و بسط پیدا می‌کنند. از پارامتر var جهت محدود کردن تغییرات چگالی مجاز داخل خوشه استفاده می‌شود. در واقع در این الگوریتم $AVGDST_C$ و DST_P تصمیم می‌گیرند که نقطه‌ای متعلق به خوشه‌ای باشد یا خیر. در این مقاله به مسائل پل نویز و شکاف خوشه‌ها نیز اشاره شده است. این دو مسئله به‌دلیل انتخاب نامناسب مقدار k رخ می‌دهند. بنابراین با انتخاب یک مقدار مناسب و دقیق برای k می‌توانیم بر این مسائل غلبه کنیم.

۱۴.۶. الگوریتم VMDBSCAN

در [۳۸] الگوریتمی دیگر برای غلبه بر مسئلهٔ تغییرات چگالی ارائه شده است. شبه کد این الگوریتم در شکل (۱۹) نشان داده شده است.

Algorithm #13: VMDBSCAN Algorithm
Input: D,Eps,Minpts, η

```
1. Begin initialize η
2. For i = 1 to n
3.         d_i ← density(x_i)
4. End For
5. Class ← DBSCAN()
6. For j = 1 to c
7.         c_j ← core(x_j )
8. End For
9. For i = 1 to n
10.        E_i = c_i − density(x_i)
11.        For j = 1 to c
12.                E_ic = c_j − density(x_i)
13.                if E_ic <E_i
14.                        vibrate the point
15.                else no vibrate
16.                End If
17.        End For
18. End For
19. End // VMDBSCAN
```

شکل (۱۹): الگوریتم VMDBSCAN

این الگوریتم ابتدا تابع چگالی هر نقطه را به دست می‌آورد. سپس با اعمال DBSCAN بر روی مجموعه داده، مرکز هر خوشه را به دست می‌آورد. سپس به‌ازای هر شیء، اگر تابع چگالی کلی آن شیء با توجه به مرکز خوشه‌ای که متعلق به آن است، بیشتر از تابع چگالی کلی با توجه به مرکز خوشه‌های دیگر باشد، آن نقطه را به‌سمت خوشه‌ای که شیء مرکزی آن حداکثر تأثیر را بر روی آن شیء دارد، حرکت می‌دهیم.

این الگوریتم نسبت به DBSCAN تعداد صحیح‌تری از خوشه‌ها را تشخیص می‌دهد، هرچند که برای داده‌های با بُعد بالا، همچنان تعداد خوشه‌های ناصحیح قابل‌توجه است. همچنین پیچیدگی زمانی این الگوریتم نسبت به DBSCAN بیشتر است.

۱.۱۴.۶. کارهای آتی

تمرکز کارهای آتی این الگوریتم می‌تواند بر روی تعیین بهترین مقدار برای پارامتر η (نرخ یادگیری) و همچنین بهبود نتایج برای مجموعه داده‌های با بعد بالا باشد.

۱۵.۶. الگوریتم Incremental DBSCAN

در [۳۹] یک الگوریتم افزایشی از DBSCAN ارائه شده است. این الگوریتم به‌جای اینکه نقاط را به‌صورت افزایشی اضافه کند، خوشه‌ها را به‌صورت افزایشی اضافه می‌کند. شبه کد این الگوریتم در شکل (۲۰) نشان داده شده است.

این الگوریتم ابتدا مجموعه نقاطی را که به‌تازگی به پایگاه داده افزوده شده‌اند، با استفاده از DBSCAN خوشه‌بندی می‌کند. سپس نقاط تقاطع، یعنی نقاطی که در مجموعه دادهٔ جدید اضافه‌شده وجود دارند ولی در مجموعه داده قبلی وجود ندارند، منتها حداقل یک نقطه در همسایگی آن نقطه در مجموعه دادهٔ قدیم وجود دارد و برعکس، مشخص می‌شوند. بعد از افزودن خوشه‌های جدید به خوشه‌های موجود در پایگاه داده، ممکن است اعضای برخی از خوشه‌ها تغییر کنند یا برخی خوشه‌ها با هم ادغام شوند یا حتی خوشه‌های جدید ایجاد شود. بنابراین ما برای هریک از نقاط تقاطع از الگوریتم ارائه‌شده در شکل (۲۰) استفاده می‌کنیم تا وضعیت نهایی خوشه‌ها مشخص شود.

Algorithm #14: Incremental DBSCAN Algorithm
Input: Old RTree, New RTree, Point

```
1. Get the Neighborhood Points of the Point in Both the RTree
2. For Every Neighborhood Point of the Given Point
3.    If the Neighborhood Point is a core Point
4.      If the Neighborhood Point was not a Core Point earlier
5.         If the Neighborhood Point belongs to a Cluster
6.           Add the Neighborhood point to a List 'Change' to process Later
7.         Else
8.           Mark to Change the Cluster Later
9.         Get the Neighborhood Points of the Neighborhood Point and add them into a list Np
10.        For every Point in the List Np
11.         If the Point is Noise
12.           Mark to Change Later
13.         If the Point Belongs to a Cluster
14.           Add the point to the List 'Change' to process Later
15.   Else
16.     Add the Neighborhood Point to the List 'Change' to process Later
17. IF No New Core Points
18.    Assign the Cluster ID as Noise
19. Else
20.    IF 'Change' List has No elements Then
21.      Assign all Marked Points to a New Cluster
22.    Else If Change List has Only one Element
23.      Add the Point to the 'Change' Cluster
24.    Else
25.      Update All the CusterID
26. End // IncrementalDBSCAN
```

شکل (۲۰): الگوریتم Incremental DBSCAN

بر اساس الگوریتم ارائه‌شده در شکل (۲۰)، چهار حالت برای یک نقطهٔ تقاطع ممکن است رخ دهد:

- حالت ۱: در صورتی که نقطهٔ تقاطع در مجموعه دادهٔ جدید یک نقطهٔ نویز باشد، نقاط تأثیرپذیر از این نقطه، یعنی نقاطی را که در همسایگی شعاع Eps این نقطه هستند بعلاوهٔ نقاطی که حداقل از یکی از نقاط موجود در همسایگی شعاع Eps این نقطه دسترسی‌پذیر چگالی هستند، مورد بررسی قرار می‌دهیم تا ببینیم آیا نقطهٔ تقاطع از هیچ نقطهٔ مرکزی موجود در مجموعه دادهٔ قدیم دسترسی‌پذیر چگالی است. در صورتی که اینچنین باشد، اعضای آن خوشه تغییر خواهد کرد. اگر نقطهٔ تقاطع با احتساب اعضای خوشه‌های قدیمی تبدیل به یک نقطهٔ مرکزی شود، آنگاه یک خوشهٔ جدید ایجاد می‌شود.
- حالت ۲: در صورتی که نقطهٔ تقاطع در مجموعه داده جدید یک نقطهٔ مرکزی باشد، ما نقاط تأثیرپذیر از آن را بررسی می‌کنیم. در صورتی که این نقطهٔ تقاطع از یک نقطهٔ مرکزی موجود در مجموعه دادهٔ قدیم دسترسی‌پذیر چگالی باشد آنگاه خوشه‌ها با هم ادغام می‌شوند.
- حالت ۳: در صورتی که نقطهٔ تقاطع در مجموعه داده جدید یک نقطهٔ حاشیه‌ای باشد، اگر این نقطه تبدیل به یک نقطهٔ مرکزی شود آنگاه خوشه‌ها با هم ادغام می‌شوند. در غیر این صورت نقطه متعلق به همان خوشهٔ قبلی می‌ماند.
- حالت ۴: در صورتی که نقطهٔ تقاطع در مجموعه دادهٔ قدیم یک نقطهٔ نویز باشد، اگر این نقطه تبدیل به یک نقطهٔ حاشیه‌ای از یک خوشهٔ جدید شود، این نقطه جذب آن خوشه می‌شود. در صورتی که نقطهٔ تقاطع تبدیل به یک نقطهٔ مرکزی شود آنگاه ایجاد یک خوشهٔ جدید و یا ادغام دو خوشه ممکن است رخ دهد.

این الگوریتم برای محیط‌های انبار داده (Data Warehousing) که در آن‌ها به‌صورت دوره‌ای حجمی از داده‌ها به داده‌های موجود افزوده می‌شوند، مفید است. این الگوریتم به ما اجازه می‌دهد که الگوی خوشه‌بندی داده‌های جدید را همراه با الگوی خوشه‌های موجود مشاهده کنیم. همچنین در این الگوریتم خوشه‌ها می‌توانند با یکدیگر ادغام شوند. شایان ذکر است که این الگوریتم به‌خصوص در مواقعی که تعداد نقاط تقاطع و مقدار نویز کم باشد، نسبت به دیگر الگوریتم‌های افزایشی موجود مقیاس‌پذیرتر است و نیاز به پرس‌وجوی ناحیه‌ای کمتری دارد.

۱.۱۵.۶. کارهای آتی

حذف خوشه‌ها به‌صورت افزایشی از مجموعه خوشه‌های موجود، یک کار چالش‌برانگیز است. در انبار داده، بسیاری از مواقع هنگام افزودن داده‌های جدید، برخی داده‌های قدیمی حذف می‌شوند. بنابراین مطلوب است اگر ما بتوانیم خوشه‌هایی را از مجموعه خوشه‌های موجود حذف کنیم و تأثیرش را بر روی خوشه‌های موجود ببینیم.

۱۶.۶. الگوریتم VDSC

در [۴۰] یک روش خوشه‌بندی مؤثر ارائه شده است که قابلیت تشخیص خوشه‌های تودرتو در فضای با چگالی متفاوت را دارد. شبه کد این الگوریتم در شکل (۲۱) نشان داده شده است.

Algorithm #15: VDSC Algorithm
Input: D, Minpts, α

```
1. Initally all data points (SetOfPoints) are unclassified.
2. Calculate core distance (εc) of each data points and initially all data points are unclassified.
3. Calculate the Average Core-Distance (avg_CD) of core-distance.
4. For i from 1 to N
5.   Min=9999;
6.   For i from 1 to N
7.   Point =SetOfPoints.get(i);
8.   If (Point.getCore() < min && Point.Classified== -1)
//Point.getCore() returns the core distance of Point.
9.     Min=Point.getCore();
10.  If(Point.Classified == -1) then
11.    Point.Neighborset = get_neighbors();
// get_neighbors() returns the neighbors of Point.
12.    Assign Point.Classified=0;
13.    Assign Point.ClusterID = cl_id;
14.    Assign the cl_id to all Point.Neighborset.
15.    σ = calculate();
//claculate() returns the Extended Core Distance
16.    Expand_cluster(SetOfPoints,Point.Neighborset, Minpt, σ )
17.  End If
18.  cl_id++;
19. End For
20. End // VDSC
// Expand_Cluster Function
Expand_cluster(SetOfPoints,Neighborset,Minpt, σ )
21. count = Neighborset.size();
22. For i from 1 to Neighborset.size
23.   Point = Neighborset.get(i);
24.   If(Point.Classified == -1)
25.   Point.Classified=0;
26.   For j from 1 to N do
27.     dist=Calculate_dist(Point,j, σ ,Neighborset);
28.     If dist != 0
29.       X.add(dist);
30.     End if
31. End For
32. If(X.size >= Minpt)
33.   Assign cl_id to all points in Neighborset;
34. End If
35. End If
36. End For
37. If count = Neighborset.size()
38.   Assign NoiseID to all points having cl_id;
39.   cl_id = cl_id -1;
40. End If
41. End // Expand_cluster
```

شکل (۲۱): الگوریتم VDSC

در این الگوریتم ابتدا فاصله از مرکز (ε) هر نقطه و میانگین فاصله از مرکزها محاسبه می‌شوند. فاصله از مرکز یک شیء مثل p عبارت است از حداکثر فاصله‌ای که شرط Minpts را ارضا کند. سپس شیء آغازگر، یعنی شیئی که حداقل فاصله از مرکز را داشته باشد، تعیین می‌شود. فرایند خوشه‌بندی با شیء آغازگر شروع می‌شود. سپس همهٔ همسایه‌های شیء آغازگر که در فاصلهٔ ε از آن قرار دارند بازیابی می‌شوند. آغازگر و همسایه‌هایش برچسب خوشه می‌گیرند و سپس فاصله از مرکز بسط‌یافته (σ) به‌صورت زیر محاسبه می‌شود:

$$\sigma = \varepsilon_c + (\alpha * avg\_CD) \qquad (۳)$$

در رابطهٔ بالا ε_c معادل با فاصله از مرکز خوشهٔ بسط‌یافته، α معادل با حد آستانهٔ تغییرات چگالی و avg_CD معادل با میانگین فاصله از مرکز است. سپس فرایند بسط خوشه با همسایه‌های آغازگر شروع می‌شود. این همسایه‌ها به یک لیست با عنوان Neighborset وارد می‌شوند و تعداد اشیاء موجود در این لیست (Count) نیز نگهداری می‌شود. در طول فرایند بسط، همهٔ اشیاء موجود در همسایگی σ از همسایه‌های شیء آغازگر بازیابی می‌شوند. شیء p در همسایگی σ از شیء q است. اگر که اولاً شیء p داخل فاصلهٔ σ از شیء q باشد و ثانیاً شیء q نیز متعلق به همسایه‌های قرارگرفته در فاصلهٔ ε از یک شیء آغازگر باشد. اگر تعداد کلی اشیاء قرارگرفته در همسایگی σ بزرگ‌تر یا مساوی Minpts باشند، این اشیاء برچسب خوشهٔ مشابه با برچسب خوشهٔ شیء آغازگر می‌گیرند. این فرایند برای همهٔ اشیاء قرارگرفته در Neighborset تکرار می‌شود. زمانی که هیچ شیئی نتواند به Neighborset اضافه شود، تعداد اشیاء موجود در Neighborset محاسبه می‌شوند. اگر این تعداد با Count برابر باشند، همهٔ اشیاء با برچسب خوشهٔ آغازگر به‌عنوان نویز لحاظ می‌شوند. در غیر این صورت ما یک خوشهٔ طبیعی با برچسب خوشهٔ مشابه با برچسب خوشهٔ شیء آغازگر داریم. کل این فرایند با انتخاب شیء آغازگر بعدی تکرار می‌گردد تا خوشهٔ بعدی پیدا شود. فرایند خوشه‌بندی تا زمانی ادامه

می‌یابد که همهٔ اشیاء برچسب خوشه یا نویز داشته باشند.

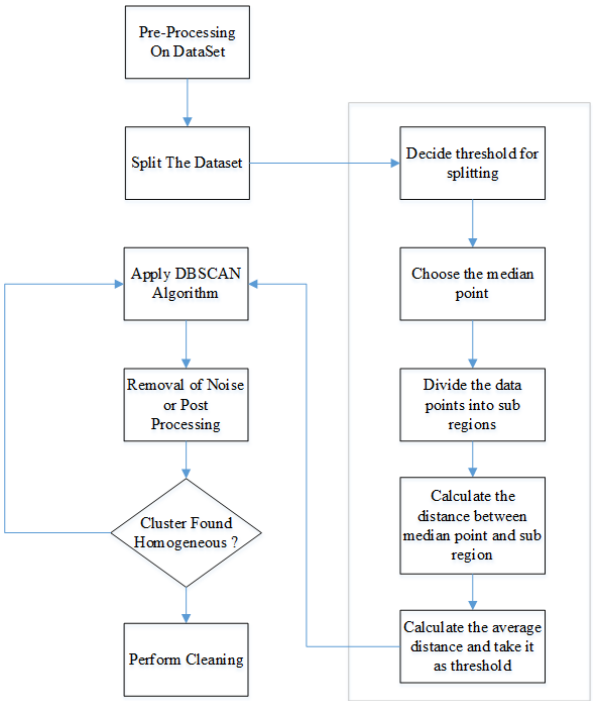
یکی از مزایای این الگوریتم تعداد کم پارامترهای ورودی آن است. پیچیدگی زمانی این الگوریتم مشابه با پیچیدگی زمانی الگوریتم DBSCAN است، اما برخلاف الگوریتم DBSCAN، این الگوریتم قابلیت تشخیص خوشه‌های تودرتو در فضای با چگالی متفاوت را دارد. این الگوریتم کارایی بهتری نسبت به DVBSCAN دارد، ولی نسبت به VDBSCAN نتوانسته کارایی بالاتری ارائه دهد.

۱.۱۶.۶. کارهای آتی

از جمله کارهای آتی مربوط به این الگوریتم می‌توان به کاهش پیچیدگی زمانی این الگوریتم و همچنین محاسبهٔ پویا مقدار پارامتر α (مقدار تغییرات چگالی مجاز) برای هر مجموعه داده‌ای اشاره کرد.

۱۷.۶. الگوریتم Improved DBSCAN

در [۴۱] بهبودی از الگوریتم DBSCAN ارائه داده است که قابلیت تشخیص خوشه‌های تودرتو را دارد. این الگوریتم قبل از شکل‌دهی خوشه‌ها، مجموعه داده را بر مبنای یک حد آستانه تقسیم می‌کند. نمودار جریان این الگوریتم در شکل (۲۲) نشان داده شده است.



شکل (۲۲): نمودار جریان الگوریتم Improved DBSCAN

قبل از اعمال الگوریتم DBSCAN انجام پیش‌پردازش ضروری است. در هنگام انجام پیش‌پردازش، مقادیر ازدست‌رفته در مجموعه داده باید مدیریت شوند و اگر دادهٔ نویز در مجموعه داده دیده شود، باید حذف گردد.

مرحلهٔ بعدی تقسیم کردن مجموعه داده است. عمل تقسیم مجموعه داده تا زمانی ادامه می‌یابد که ما به یک حد آستانه برسیم. تعیین دقیق این حد آستانه بسیار مهم است. اگر این حد آستانه را خیلی بزرگ در نظر بگیریم، دقت تقسیم‌بندی کاهش می‌یابد و اگر هم که آن را کوچک بگیریم، نواحی خیلی کوچک ایجاد می‌شوند. ازاین‌رو برای به‌دست‌آوردن حد آستانه به این صورت عمل می‌کنیم که ابتدا نقطهٔ میانی مجموعه داده را انتخاب می‌کنیم و سپس مجموعه داده را به زیرنواحی تقسیم می‌کنیم. سپس فاصلهٔ اقلیدسی بین مقدار میانه و هریک از زیرنواحی مجموعه داده را محاسبه می‌کنیم و فاصله‌های به‌دست‌آمده را مرتب می‌کنیم. آنگاه میانگین فواصل از همهٔ زیرنواحی را به‌عنوان حد آستانه برای تقسیم کردن مجموعه داده در نظر می‌گیریم.

بعد از تقسیم کردن مجموعه دادهٔ الگوریتم DBSCAN را اعمال می‌کنیم. بعد از اعمال الگوریتم DBSCAN و پیدا کردن خوشه‌ها، ممکن است بعضی از نقاط هنوز خوشه‌بندی نشده باشند یا به‌عنوان نویز یا نقطهٔ حاشیه‌ای در نظر گرفته شده باشند. همچنین ممکن است برخی از خوشه‌ها بسیار کوچک باشند. ازاین‌رو نیاز به عمل پس‌پردازش است. طی عمل پس‌پردازش، نواحی نویز با نزدیک‌ترین خوشه ادغام می‌شوند.

هدف بعدی بررسی داخل خوشه‌ها و شناسایی زیربخش‌هاست. برای این منظور هر دو مقدار Eps و Minpts کمی افزایش می‌یابند و دوباره DBSCAN فراخوانی می‌شود. در پایان نیز هر قسمتی که به‌عنوان نویز وجود داشته باشد با نزدیک‌ترین خوشه ادغام می‌شود.

تعداد خوشه‌های یافت‌شده توسط این الگوریتم نسبت به الگوریتم DBSCAN بیشتر است. همچنین این مقاله یک بررسی موردی در مورد معیار فاصلهٔ مورد استفاده الگوریتم‌های خوشه‌بندی انجام داده است. بر اساس نتایج

به‌دست‌آمده در این مقاله، معیار فاصلهٔ اقلیدسی برای داده‌های با بعد پایین خوب کار می‌کند، ولی برای داده‌های با بعد بالا خیر. همچنین در این مقاله نشان داده شده است که فاصلهٔ اقلیدسی در مقایسه با فاصلهٔ مانهاتان (Manhattan Distance) تعداد خوشهٔ بیشتری را تشخیص می‌دهد، ولی از طرفی تعداد نمونه‌هایی که به‌صورت اشتباه خوشه‌بندی شده‌اند نیز بیشتر است. همچنین نرخ نویز نیز در زمان استفاده از فاصلهٔ مانهاتان نسبت به فاصلهٔ اقلیدسی بیشتر است.

۱.۱۷.۶. کارهای آتی

در این الگوریتم ممکن است برخی از نقاط مرکزی از دست بروند که این مسئله باعث از دست رفتن اشیاء می‌شود. به‌عنوان کارهای آتی، می‌توان به حل این مسئله پرداخت.

۱۸.۶. الگوریتم Revised DBSCAN

با توجه به اینکه الگوریتم DBSCAN زمانی که خوشه‌ها نزدیک به هم باشند ممکن است با شکست مواجه شود، در [۴۲] الگوریتمی برای رفع مشکل خوشه‌های مجاور ارائه شده است. در این الگوریتم به‌جای استفاده از مفهوم دسترسی‌پذیر چگالی (Density Reachable) از مفهوم دسترسی‌پذیر چگالی مرکزی (Core Density Reachable) استفاده شده است. شبه کد این الگوریتم در شکل (۲۳) نشان داده شده است.

الگوریتم مشابه با الگوریتم DBSCAN است با این تفاوت که ابتدا خوشه‌های متشکل از اشیاء مرکزی پیدا می‌شوند و سپس اشیاء حاشیه‌ای به نزدیک‌ترین شیء مرکزی تخصیص داده می‌شوند. در واقع در این الگوریتم در زنجیره دسترسی‌پذیری چگالی بهبود انجام‌شده است و این زنجیره فقط شامل اشیاء مرکزی است و اشیاء حاشیه‌ای به یک لیست مجزا اضافه می‌شوند. در انتها هریک از اشیاء حاشیه‌ای به نزدیک‌ترین مرکز تخصیص داده می‌شوند (مراحل ۱۰ــ۱۳ از شکل ۲۳). بنابراین همان طور که در مراحل ۲۱ـ۲۳ از شکل (۲۳) مشاهده می‌کنید، الگوریتم بهبودیافته، برخلاف الگوریتم DBSCAN فقط نقاط مرکزی را برچسب خوشه می‌زند و تصمیم‌گیری در مورد اشیاء حاشیه‌ای بعد از شکل‌دهی خوشه‌های متشکل از اشیاء مرکزی، یعنی در مراحل ۱۰ـ۱۳ از شکل (۲۳) انجام می‌شود. این الگوریتم بر روی هر دوی داده‌های مکانی و غیرمکانی قابل اعمال است.

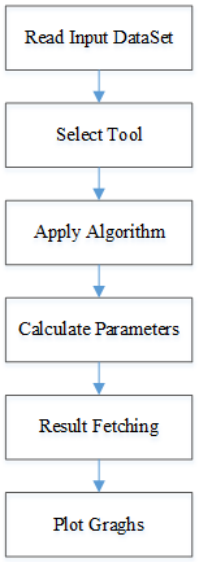
Algorithm #16: Revised DBSCAN Algorithm
Input: D, Eps, MinPts

1. ClusterId=1
2. For x_i in data set
3. If x_i is UNCLASSIFIED
4. ExpandCoreCluster(x_i,ClusterId)
5. If ExpandCoreCluster successful
6. ClusterId=ClusterId+1
7. End If
8. End If
9. End For
10. For x_{border} in the border list
11. $N_{Eps}(x_{border})$=Retrieve_Neighbors(x_{border},Eps)
12. Assign x_{border} to ClusterId of the closest core object in $N_{Eps}(x_{border})$
13. End For
14. End // Revised DBSCAN
// ExpandCoreCluster Function
ExpandCoreCluster(x_i,ClusterId)
15. Seeds=Retrieve_Neighors(x_i,Eps)
16. If |seeds|<Minpts
17. Mark x_i as noise
18. Return without expansion success
19. Else
20. For all x_j in seeds list
21. $N_{Eps}(x_j)$=Retrieve_Neighors(x_j,Eps)
22. If $|N_{Eps}(x_j)| \geq$ Minpts // x_j is a core object
// further cluster expansion only for a core object
23. Assign x_j to ClusterId
24. Add all UNCLASSIFIED in $N_{Eps}(x_j)$ to seeds list
25. Lable UNCLASSIFIED and noise objects in $N_{Eps}(x_j)$ as BORDER objects
// Note : the noise within the neighborhood of a core object is labeled as a border object
26. End If
27. End For
28. Return with expansion success
29. End If
30. End // ExpandCoreCluster

شکل (۲۳): الگوریتم Revised DBSCAN

۱۹.۶. الگوریتم Improvement of DBSCAN

در [۴۳] الگوریتمی ارائه شده است که نسبت به الگوریتم DBSCAN مقیاس‌پذیرتر است. این الگوریتم به‌جای کار بر روی کل پایگاه داده بر روی قسمت‌های تقسیم‌شده از آن کار می‌کند. فلوچارت مربوط به این الگوریتم در شکل (۲۴) نشان داده شده است.



شکل (۲۴): فلوچارت الگوریتم Improvement of DBSCAN

همان گونه که در شکل (۲۴) مشاهده می‌کنید، به‌طور کلی روال انجام کار این الگوریتم شامل مراحل زیر است.

۱. خواندن داده‌ها از مجموعه داده: در اینجا از انواع مختلفی از مجموعه داده‌ها همراه با ویژگی‌های متعدد استفاده شده است. در ابتدا همهٔ این مجموعه داده‌ها به مجموعه داده‌های مختلف شکسته می‌شوند.

۲. انتخاب ابزار: بعد از خواندن مجموعه داده‌ها، مرحلهٔ بعدی از الگوریتم پیشنهادی، انتخاب ابزار داده‌کاوی Weka است.

۳. اعمال الگوریتم DBSCAN بر روی داده‌ها: زمانی که همه مجموعه داده‌ها در ابزار Weka بارگذاری شدند، آنگاه الگوریتم DBSCAN بر روی مجموعه داده‌های مختلف اعمال می‌شود.

۴. محاسبهٔ پارامترها: قبل از اعمال الگوریتم DBSCAN بر روی مجموعه داده، کاربر باید پارامترهای مورد نیاز را تنظیم کند.

۵. ارزیابی نتایج: در این مرحله، کارایی مورد ارزیابی قرار می‌گیرد و نتایج مجموعه داده‌های مختلف با هم مقایسه می‌شوند. این مقایسه بر اساس معیارهایی نظیر تعداد خوشه‌های یافت‌شده، تعداد نمونه‌های به‌اشتباه خوشه‌بندی‌شده، زمان خوشه‌بندی و آنالیز نویز صورت می‌گیرد.

۶. نمایش نتایج ارزیابی به‌صورت گرافیکی.

با توجه به اینکه این الگوریتم بر روی قسمت‌های تقسیم‌شده از مجموعه داده کار می‌کند، نسبت به الگوریتم DBSCAN مقیاس‌پذیرتر است و می‌تواند برای مجموعه داده‌های بزرگ کارایی خوبی از خود نشان دهد. الگوریتم پیشنهادی تعداد خوشه‌های بیشتری را نسبت به DBSCAN تشکیل می‌دهد و ازاین‌رو از دقت بالاتری برخوردار است.

۲۰.۶. An enhancement of DBSCAN

یک نسخهٔ بهبودیافته دیگر از الگوریتم DBSCAN در [۴۴] ارائه شده است که قابلیت تولید نتایج مناسب برای مجموعه داده‌های خیلی بزرگ را دارد. در واقع هدف این الگوریتم افزایش مقیاس‌پذیری و کاهش پیچیدگی زمانی الگوریتم DBSCAN است. برای رسیدن به این اهداف، الگوریتم طی یک مرحلهٔ پیش‌پردازش، مجموعه داده را با استفاده از CLARANS [۴۵] تقسیم‌بندی می‌کند. این تقسیم‌بندی مجموعه داده باعث می‌شود که تلاش برای یافتن اشیاء مرکزی کاهش یابد، چراکه به‌جای بررسی کل مجموعه داده فقط نواحی تقسیم‌بندی‌شده مورد بررسی قرار می‌گیرند. سپس نواحی متراکم به‌دست‌آمده از هریک از نواحی تقسیم‌شده با یکدیگر ادغام می‌شوند. عمل ادغام با استفاده از «اتصال داخلی نسبی» تعریف‌شده در [۴۶] انجام می‌شود.

به‌طور کلی نقاط قوت این الگوریتم عبارت‌اند از: ۱. با توجه به محدود کردن فضای جستجو به نواحی تقسیم‌بندی‌شده به‌جای کل مجموعه داده، این الگوریتم نیاز به زمان کمتری نسبت به DBSCAN برای خوشه‌بندی مجموعه داده دارد؛ ۲. میزان حافظهٔ کارآمد به‌دلیل اندازهٔ بافر کوچک مورد استفادهٔ الگوریتم. این الگوریتم نیز همانند الگوریتم پایهٔ DBSCAN نیاز به دو پارامتر ورودی Eps و Minpts دارد.

۲۱.۶. الگوریتم O-VDBSCAN

ایراد اصلی الگوریتم VDBSCAN، تعیین پارامتر k برای محاسبهٔ k-dit Plot است. در [۴۷] بهبودی از الگوریتم

VDBSCAN ارائه شده است که در آن مقدار k به‌صورت خودکار محاسبه می‌شود. شبه کد این الگوریتم در شکل (۲۵) نشان داده شده است.

Algorithm #17: O-VDBSCAN Algorithm
Input: D

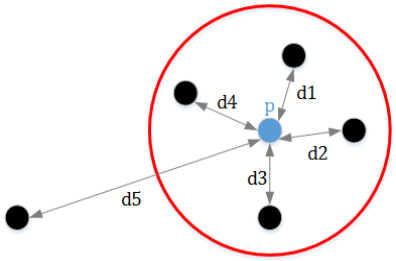
1. Calculates and stores k-dists of all objects for every k in a given domain (initial domain).
2. Calculates the overall change rate of k-dist:

$$\Delta d_k = \sum_{J=1}^{n} (d_{j,k+1} - d_{j,k})$$

3. Finds the first that exceeds the defined threshold.
4. Apply VDBSCAN with the k selected in (3) on the dataset.

شکل (۲۵): الگوریتم O-VDBSCAN

با توجه به شکل (۲۶)، فرض کنید d_k فاصلهٔ شیء p از kاُمین نزدیک‌ترین همسایه‌اش باشد. برای مثال d_1 فاصله شیء p از اولین نزدیک‌ترین همسایه‌اش است. همچنین فرض کنید Δd_k یا نرخ تغییر برابر با d_{k+1}-d_k باشد. همان گونه که در شکل (۲۶) می‌بینید، Δd_1، Δd_2، Δd_3 و Δd_4 اختلاف چندانی با هم ندارند. اما با توجه به اینکه تفاضل Δd_5 و Δd_4 از تفاضل Δd_4 و Δd_3 افزایش زیادی داشته، به احتمال زیاد چهارمین نزدیک‌ترین همسایه بر روی مرز این خوشه قرار داشته و ازاین‌رو بهترین مقدار برای پارامتر k (Minpts) برابر با ۵ خواهد بود. الگوریتم O-VDBSCAN با محاسبهٔ نرخ تغییر کلی برای یک دامنه از مقادیر k (مرحلهٔ ۲ از شکل ۲۵)، اولین k را که از یک حد آستانه تجاوز کند، به‌عنوان مقدار نهایی برای پارامتر k در نظر می‌گیرد. سپس الگوریتم VDBSCAN با استفاده از این k، مقادیر مختلف Eps را محاسبه کرده و در نهایت الگوریتم DBSCAN به‌ازای هر Eps به‌دست‌آمده بر روی مجموعه داده اعمال می‌شود تا خوشه‌های موجود در مجموعه داده شناسایی شوند.



شکل (۲۶): مثالی برای نشان دادن رابطه بین پارامتر k و نرخ تغییر

همان‌گونه که دیدیم، الگوریتم O-VDBSCAN برای محاسبهٔ مقدار مناسب برای پارامتر k مربوط به الگوریتم VDBSCAN، از یک دامنهٔ اولیه برای این پارامتر استفاده می‌کند که استفاده از این روش چندان کارآمد نیست؛ چراکه برای مجموعه داده‌های مختلف این دامنه تغییر می‌کند. در نتیجه خود تعیین این دامنه یک مسئلهٔ جدید است.

۱.۲۱.۶. کارهای آتی

با توجه به بزرگ بودن دامنهٔ اولیهٔ k، محدود کردن دامنهٔ اولیه پارامتر k به‌منظور انجام محاسبات کمتر یکی از کارهای پیش روی محققان است.

۲۲.۶. الگوریتم PDBSCAN

همان گونه که در بخش قبلی نیز اشاره کردیم، یکی از تکنیک‌های موجود برای غلبه بر پیچیدگی زمانی الگوریتم DBSCAN، موازی‌سازی این الگوریتم است. در [۴۸] یک نسخهٔ موازی از الگوریتم DBSCAN ارائه شده است. برای پیاده‌سازی این الگوریتم از معماری Shared-Nothing استفاده شده است. یکی از مباحث مهم در هنگام موازی‌سازی یک الگوریتم خوشه‌بندی بر روی معماری Shared-Nothing ساختار دادهٔ توزیع‌یافتهٔ مورد استفاده در آن است. نویسندهٔ این مقاله برای رسیدن به یک استراتژی قراردهی دادهٔ مناسب، یک ساختار دادهٔ توزیع‌شده به نام dr*-tree را معرفی کرده است. تفاوت بین dr*-tree و r*-tree معمولی در این است که در dr*-tree صفحات داده برخلاف r*-tree بر روی کامپیوترهای مختلف پخش شده‌اند. همچنین در این ساختار هر نود علاوه بر اطلاعات قبلی شامل یک اشاره‌گر به کامپیوتری که اطلاعات موجود در فرزندان آن نود در آن قرار دارند، نیز می‌باشد. نتایج نشان داده است که این الگوریتم توانسته به Speed-Up و Scale-Up مناسب و قابل‌توجهی برسد.

همچنین در [۴۹ و ۵۰] نیز روش‌های دیگری جهت موازی‌سازی الگوریتم DBSCAN ارائه شده است.

۱.۲۲.۶. کارهای آتی

در این مقاله، ساختار شاخص r*-tree به‌منظور استفاده در یک محیط توزیع‌شده بسط داده شده است. می‌توان به روش مشابه،

سایر ساختارهای شاخص مکانی مانند x-tree [۵۱] را نیز برای استفاده در محیط‌های توزیع‌شده بسط داد. همچنین می‌توان الگوریتم DBSCAN را روی سایر معماری‌های موازی‌سازی مانند معماری Shared-Memory پیاده‌سازی کرد.

علاوه بر توازن بار، یکی دیگر از ویژگی‌های بسیار مهم در یک استراتژی قراردهی داده، کاهش هزینهٔ ارتباطات است. برای کاهش هزینهٔ ارتباطات، داده‌ها باید به‌گونه‌ای تقسیم شوند که داده‌های نزدیک به هم بر روی یک کامپیوتر قرار بگیرند. این مقاله با اعمال منحنی هیلبرت بر روی صفحات دادهٔ موجود در نودهای برگ r*-tree سعی در رسیدن به این هدف کرده است. اما اعمال منحنی هیلبرت بعد از ساخت r*-tree با توجه به اینکه صفحات دادهٔ r*-tree می‌توانند همپوشانی داشته باشند، چندان معقول نیست. به همین دلیل، پیشنهاد می‌کنیم که از ساختار شاخص Hillbert r-tree [۵۲] استفاده شود. تفاوت این ساختار با ساختار استفاده‌شده در مقاله، در زمان اعمال منحنی هیلبرت است. در Hillbert r-tree منحنی هیلبرت قبل از ساخت درخت اعمال می‌شود و r-tree از همان ابتدا بر اساس مقادیر هیلبرت ساخته می‌شود.

۷. مقایسهٔ الگوریتم‌های با قابلیت تشخیص خوشه‌های با چگالی متفاوت

در این قسمت ضمن اینکه مقایسه‌ای بین چند الگوریتم خوشه‌بندی مبتنی بر چگالی با قابلیت تشخیص خوشه‌های با چگالی متفاوت انجام می‌شود، با روش‌های ارزیابی الگوریتم‌های خوشه‌بندی نیز آشنا می‌شویم.

با توجه به تعداد زیاد الگوریتم‌های خوشه‌بندی که برای حل مشکل تغییرات چگالی الگوریتم DBSCAN ارائه شده‌اند، انتخاب بهترین الگوریتم کاری چالش‌برانگیز است. ازاین‌رو با توجه به مطالعات انجام‌گرفته، ما پنج الگوریتم VDBSCAN، LDBSCAN، DVBSCAN، MDDBSCAN و VMDBSCAN را انتخاب کرده و اقدام به مقایسهٔ آن‌ها کرده‌ایم. به‌منظور ارزیابی و مقایسه، این پنج الگوریتم با استفاده از زبان برنامه‌نویسی سی‌شارپ در محیط ویندوز پیاده‌سازی شده‌اند و از چهار مجموعه دادهٔ استاندارد [۵۳] PathBased، Jain، Aggregation و Can473 و همچنین دو مجموعه دادهٔ مصنوعی برای ارزیابی استفاده شده است. اطلاعات مربوط به مجموعه داده‌های مورد استفاده برای ارزیابی در جدول (۱) نشان داده شده است.

جدول (۱): مجموعه داده‌های مورد استفاده

نام مجموعه‌داده	نوع مجموعه‌داده	تعداد اشیاء	تعدادخوشه	چندچگالی
Aggregation	Standard	788	7	No
Jain	Standard	373	2	Yes
Can473	Standard	473	8	Yes
Path-Based	Standard	300	3	No
DataSet 1	Artificial	10208	9	Yes
DataSet 2	Artificial	2671	10	Yes

به‌منظور مقایسهٔ این الگوریتم‌ها از سه معیار Rand [۵۴]، Jaccard [۵۵] و Fowlkes-Mallows [۵۶] که از جمله معیارهای ارزیابی خوشه‌بندی هستند، و معیار درصد خطا خوشه‌بندی استفاده کرده‌ایم. سه معیار اول میزان شباهت خوشه‌های حاصل از الگوریتم‌های مورد ارزیابی با خوشه‌های برچسب‌دار را محاسبه می‌کنند. این معیارها از فرمول‌های زیر به دست می‌آیند:

$$\text{Rand-Measure}=\frac{TP+TN}{TP+FP+FN+TN} \qquad (۴)$$

$$\text{Jaccard-Index}=\frac{|A\cap B|}{|A\cup B|}=\frac{TP}{TP+FP+FN} \qquad (۵)$$

$$\text{Fowlkes Mallows-Index}=\sqrt{\frac{TP}{TP+FP}\cdot\frac{TP}{TP+FN}} \qquad (۶)$$

همهٔ این فرمول‌ها بر مبنای روش‌های بیان خطا False Positive(FP)، False Negative(FN) و True Positive(TP) و True Negative(TN) بیان شده‌اند. با برداشتن دو شیء از مجموعه داده، TP زمانی برقرار است که این دو شیء واقعاً متعلق به یک خوشه باشند (بر طبق مجموعه داده برچسب خورده) و الگوریتم مورد ارزیابی نیز آن دو شیء را به یک خوشه تخصیص داده باشد. اگر این دو شیء واقعاً متعلق به یک خوشه نباشند و الگوریتم مورد ارزیابی هم آن‌ها را به یک خوشه تخصیص نداده باشد، TN رخ داده است. در صورتی که این دو شیء واقعاً متعلق به یک خوشه باشند و الگوریتم

مورد ارزیابی آن‌ها را به یک خوشه تخصیص نداده باشد، FN رخ داده است و نهایتاً در صورتی که این دو شیء واقعاً متعلق به یک خوشه نباشند و الگوریتم مورد ارزیابی آن‌ها را به یک خوشه تخصیص داده باشد، FP رخ داده است. بنابراین با چک کردن تمامی جفت نقاط مجموعه داده، این چهار معیار تعیین خطا مشخص می‌شوند.

نتایج حاصل از اعمال الگوریتم‌های مدنظر بر روی شش مجموعه دادهٔ موردنظر بر اساس معیارهای Rand، Jaccard و Fowlkes Mallows در جدول‌های (۲) و (۳) و (۴) نشان داده شده است.

جدول (۲): نتایج ارزیابی بر اساس معیار Rand

Datasets / Algorithm	Can473	Jain	Aggregation	PathBased	DS1	DS2
LD	0.876	0.967	0.964	0.925	0.97	0.98
VD	0.792	0.614	0.926	0.752	0.63	0.81
DV	0.90	0.953	0.994	0.759	0.85	0.97
VM	0.79	0.624	0.708	0.875	0.79	0.94
MD	0.98	1	0.938	0.912	0.98	1

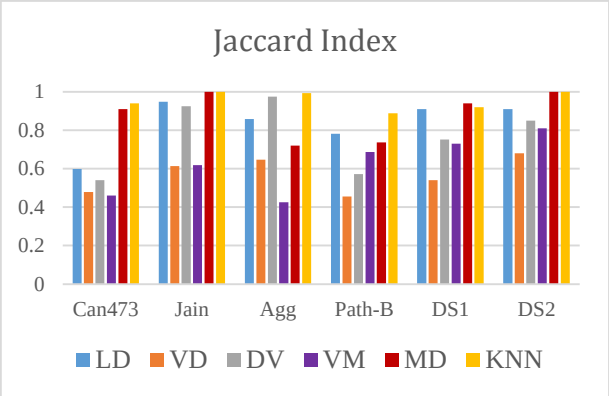
جدول (۳): نتایج ارزیابی بر اساس معیار Jaccard

Datasets / Algorithm	Can473	Jain	Aggregation	PathBased	DS1	DS2
LD	0.598	0.947	0.857	0.781	0.91	0.92
VD	0.479	0.614	0.646	0.456	0.54	0.68
DV	0.54	0.924	0.975	0.572	0.752	0.85
VM	0.46	0.618	0.425	0.686	0.73	0.81
MD	0.91	1	0.72	0.737	0.94	1

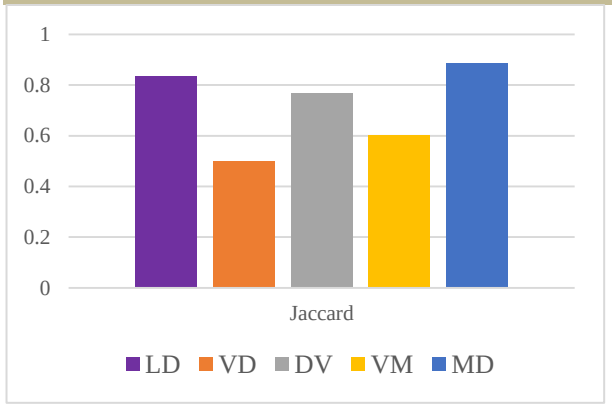
جدول (۴): نتایج ارزیابی بر اساس معیار Fowlkes Mallows

Datasets / Algorithm	Can473	Jain	Aggregation	PathBased	DS1	DS2
LD	0.767	0.973	0.924	0.88	0.94	0.985
VD	0.692	0.783	0.863	0.627	0.625	0.79
DV	0.71	0.961	0.987	0.751	0.84	0.95
VM	0.66	0.785	0.652	0.813	0.76	0.91
MD	0.95	1	0.847	0.857	0.96	1

در شکل (۲۷) نتایج به‌دست‌آمده با توجه به معیار Jaccard در قالب نمودار نشان داده شده است. همچنین، در شکل (۲۸) نیز معیار Jaccard به‌طور میانگین برای همهٔ الگوریتم‌ها نشان داده شده است. همان گونه که در شکل (۲۸) می‌بینید، الگوریتم پیشنهادی از میانگین بهتری نسبت به سایر الگوریتم‌ها برخوردار است، به این معنی که خوشه‌های حاصل از این الگوریتم شباهت بیشتری به خوشه‌های برچسب‌گذاری‌شده داشته است.



شکل (۲۷): نتایج ارزیابی با توجه به معیار Jaccard



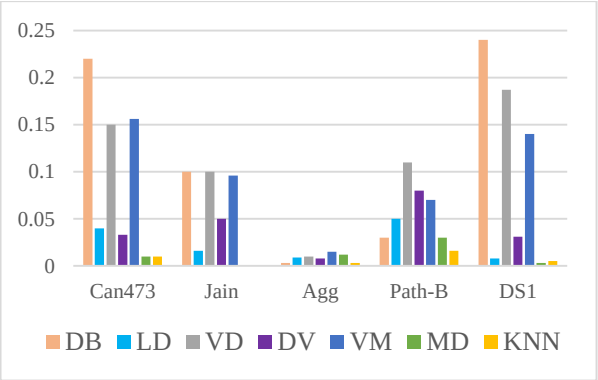
شکل (۲۸): میانگین معیار Jaccard

شاخص درصد خطا خوشه‌بندی، از تقسیم تعداد نمونه‌های به‌اشتباه خوشه‌بندی‌شده بر تعداد کل نمونه‌ها حاصل می‌شود. در جدول (۵) درصد خطا و تعداد خوشه‌های یافت‌شده توسط الگوریتم‌ها نشان داده شده است. در این جدول درصد خطای مربوط به الگوریتم DBSCAN نیز قرار داده شده است. توجه داشته باشید که تعداد خوشه‌های صحیح هر مجموعه داده در داخل پرانتز، مقابل اسم مجموعه داده مشخص شده است. در شکل (۲۹) درصد خطای مربوط به الگوریتم‌های مختلف در قالب نمودار بیان شده است. همچنین در شکل (۳۰) نمودار میانگین درصد خطا این پنج الگوریتم را مشاهده می‌کنید. همان طور که در شکل (۳۰) می‌بینید، الگوریتم MDDBSCAN به‌طور میانگین، از دقت بالاتری نسبت به سایر الگوریتم‌ها برخوردار است. همان گونه که قبلاً نیز اشاره کردیم، الگوریتم VDBSCAN تنها در مجموعه داده‌هایی که دارای شیب تند در منحنی k-dist plot متناظرشان هستند پاسخ مناسب تولید می‌کند. ازاین‌رو، همان گونه که در

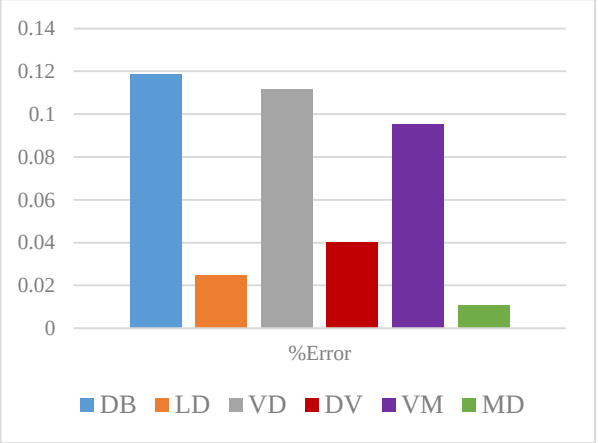
شکل (۳۰) نیز مشاهده می‌کنید، این الگوریتم درصد خطای بالاتری نسبت به سایر الگوریتم‌ها دارد. با توجه به اینکه الگوریتم DBSCAN قابلیت تشخیص خوشه‌های با چگالی متفاوت را ندارد و همچنین در تشخیص خوشه‌های نزدیک به هم دچار ضعف است، همان گونه که در شکل (۳۰) نیز مشاهده می‌کنید، درصد خطای این الگوریتم بالا است.

جدول (۵): نتایج ارزیابی بر اساس درصد خطا خوشه‌بندی

Datasets / Algorithms	Can473(8)		Jain(2)		Aggregation(7)		Path-Based(3)	
	#CTR	%err	#CTR	%err	#CTR	%err	#CTR	%err
DBSCAN	6	0.22	2	0.1	7	0.003	3	0.033
LDBSCAN	8	0.04	2	0.016	7	0.009	3	0.05
VDBSCAN	6	0.15	3	0.1	6	0.01	3	0.11
DVBSCAN	8	0.033	3	0.05	7	0.008	4	0.08
VMDBSCAN	6	0.156	2	0.096	6	0.015	3	0.07
MDDBSCAN	8	0.01	2	0	7	0.012	3	0.03



شکل (۲۹): درصد خطا الگوریتم‌های مورد ارزیابی



شکل (۳۰): میانگین درصد خطا الگوریتم‌های مورد ارزیابی

در ادامه یک دسته‌بندی و خلاصه از الگوریتم‌های ارائه‌شده در قالب جدول (۶) ارائه می‌شود.

۸. نتیجه

خوشه‌بندی یکی از زمینه‌های تحقیقاتی بسیار فعال در داده‌کاوی است. با توجه به اینکه روش‌های خوشه‌بندی مبتنی بر چگالی یکی از روش‌های پرکاربرد و اصلی برای خوشه‌بندی محسوب می‌شوند، در این مقاله ما تمرکز خودمان را بر روی الگوریتم پایه این روش‌ها یعنی الگوریتم DBSCAN گذاشته‌ایم. بعد از صحبت کردن دربارهٔ کاربردها و مزایای این الگوریتم، اقدام به بررسی مشکلات این الگوریتم کردیم و برای هریک از مشکلات مطرح‌شده، بهبودهای ارائه‌شده بر روی آن مشکل را معرفی کردیم. در واقع در این مقاله یک دسته‌بندی بسیار خوب از بهبودهای الگوریتم پرکاربرد DBSCAN و همچنین نقاط ضعف و قوت این الگوریتم‌ها ارائه شد. به‌منظور اینکه بهتر بتوانیم دربارهٔ بهبودهای ارائه‌شده صحبت و اظهار نظر کنیم، اقدام به پیاده‌سازی و همچنین مقایسه برخی از این الگوریتم‌ها کردیم. ازاین‌رو ضمن اینکه با روش‌های ارزیابی الگوریتم‌های خوشه‌بندی آشنا شدیم، توانستیم یکی از بهترین الگوریتم‌های خوشه‌بندی مورد استفاده برای محیط‌های با چگالی متفاوت را معرفی کنیم. بنابراین در انتهای مباحث، بر اساس مطالعات انجام‌شده، چند نسخه از الگوریتم DBSCAN را که برای رفع مشکل تغییرات چگالی الگوریتم DBSCAN ارائه شده بودند، انتخاب و اقدام به مقایسهٔ آن‌ها بر اساس معیارهای ارزیابی خوشه‌بندی نمودیم. بر اساس نتایج به‌دست‌آمده، الگوریتم MD-DBSCAN نسبت به سایر الگوریتم‌های مورد ارزیابی که همگی برای رفع مشکل تغییرات چگالی DBSCAN ارائه شده‌اند، از شاخص شباهت بالاتر و میزان خطا پایین‌تر برخوردار است.

در انتهای معرفی هر الگوریتم، کارهای آتی مربوط به هر الگوریتم را در صورت وجود معرفی کردیم. با این حال به‌طور کلی چندین کار را برای تحقیقات آتی پیشنهاد می‌دهیم:

❖ **تعیین خودکار پارامترهای الگوریتم‌های مبتنی بر چگالی**

به‌طور کلی، انتخاب خودکار پارامترهای الگوریتم‌های خوشه‌بندی یک کار چالش‌برانگیز است. الگوریتم‌هایی مانند

ST-DBSCAN [۲۹]، LDBSCAN [۳۰]، DVBSCAN [۳۵] و بسیاری از الگوریتم‌های دیگر نیاز دارند تا پارامترهای ورودی‌شان، به‌دلیل سخت بودن تعیین مقدار دقیق این پارامترها در مجموعه داده‌های واقعی و بزرگ، به‌طور خودکار تعیین شوند. بنابراین، ارائه روش‌هایی برای تعیین خودکار پارامترهای ورودی الگوریتم‌های مبتنی بر چگالی می‌تواند در آیندهٔ مدنظر قرار گرفته شود. برای این منظور می‌توان از روش‌های به‌کاررفته در الگوریتم VDBSCAN [۲۶] و بهبودهای آن ایده گرفت.

❖ رفع مشکل تغییرات چگالی ST-DBSCAN

الگوریتم ST-DBSCAN [۲۹] یکی از الگوریتم‌هایی پرکاربرد است که برخلاف سایر الگوریتم‌های خوشه‌بندی مبتنی بر چگالی، قابلیت تشخیص خوشه‌ها مطابق با مقادیر مکانی، غیرمکانی و زمانی اشیاء را دارد. این الگوریتم نیز همانند الگوریتم DBSCAN قابلیت تشخیص خوشه‌های با چگالی متفاوت را ندارد، ازاین‌رو بهبود این الگوریتم به‌منظور تشخیص خوشه‌های با چگالی متفاوت یکی از کارهای آتی پیش روی محققان است. می‌توان از روش‌های به‌کاررفته در الگوریتم‌هایی که برای رفع مشکل تغییرات چگالی الگوریتم DBSCAN ارائه شده‌اند ایده گرفت. منتها در اینجا با توجه به اینکه دو شعاع Eps داریم کار کمی سخت‌تر است.

❖ ارائهٔ یک الگوریتم خوشه‌بندی مبتنی بر چگالی افزایشی که توانایی کار در محیط‌های با چگالی متفاوت را داشته باشد

برای کار در محیط‌های با یک چگالی نسخهٔ افزایشی الگوریتم DBSCAN ارائه‌شده در [۳۹] کفایت می‌کند. اما وجود یک الگوریتم خوشه‌بندی مبتنی بر چگالی برای استفاده در محیط‌های با چگالی متفاوت ضروری است. در این زمینه فقط نسخهٔ افزایشی الگوریتم OPTICS در [۵۷] ارائه شده است. این الگوریتم توانایی افزودن و حذف کردن خوشه‌ها به‌صورت افزایشی را دارد. یکی از مشکلات این الگوریتم پیچیدگی بالا این الگوریتم با توجه به ماهیت الگوریتم OPTICS است. با توجه به اینکه بهبودهای بسیار زیادی از الگوریتم DBSCAN برای حل مسئلهٔ تغییرات چگالی این الگوریتم ارائه شده‌اند، ارائهٔ نسخهٔ افزایشی یکی از این بهبودها کاری است که می‌تواند در آینده مدنظر قرار گیرد. پیشنهاد ما ارائهٔ نسخهٔ افزایشی الگوریتم MD-DBSCAN [۳۷] است؛ چراکه با توجه به آزمایش‌های صورت‌گرفته، این الگوریتم یکی از بهترین الگوریتم‌های مبتنی بر چگالی با قابلیت کار در محیط‌های با چگالی متفاوت است.

❖ بسط ساختارهای شاخص مکانی برای استفاده در محیط‌های توزیع‌شده

یکی از نکات مهم هنگام موازی‌سازی الگوریتم‌های خوشه‌بندی بر روی معماری Shared-Nothing، ساختار شاخص توزیع‌شده مورد استفاده توسط آن الگوریتم است. در [۴۸] ساختار شاخص r*-tree به‌منظور استفاده در یک محیط توزیع‌شده بسط داده شده است. می‌توان به روش مشابه، سایر ساختارهای شاخص مکانی مانند x-tree [۵۱] را نیز جهت استفاده در محیط‌های توزیع‌شده بسط داد.

❖ ارائه الگوریتم‌های خوشه‌بندی مقیاس‌پذیر

یکی از روش‌های موجود برای کاهش زمان و افزایش مقیاس‌پذیری الگوریتم‌های خوشه‌بندی، موازی سازی این الگوریتم‌هاست. چندین نسخهٔ موازی از الگوریتم DBSCAN معرفی شده‌اند. با این حال هنوز وجود دارند الگوریتم‌های پرکاربرد، مانند ST-DBSCAN که نسخهٔ موازی آن‌ها ارائه نشده است و می‌توان بر روی موازی‌سازی آن‌ها کار کرد. همچنین می‌توان الگوریتم‌هایی که نسخهٔ موازی آن‌ها ارائه شده است، مانند الگوریتم DBSCAN را بر روی سایر معماری‌های موازی‌سازی پیاده‌سازی کرد.

یکی از پیشنهادات ما موازی‌سازی الگوریتم‌های خوشه‌بندی با استفاده از روش MapReduce [۵۸] است. MapReduce یک مدل برنامه‌نویسی موازی مناسب برای تجزیه‌وتحلیل مجموعه داده‌های با مقیاس بالاست که بر مبنای معماری Shared-Nothing ارائه شده است. از جمله

ویژگی‌های روش MapReduce می‌توان به سادگی، مقیاس‌پذیری و تحمل‌پذیری خطا اشاره کرد. در [۵۹] نسخهٔ موازی الگوریتم DBSCAN با استفاده از روش MapReduce ارائه شده است. به‌طور مشابه می‌توان به موازی‌سازی سایر الگوریتم‌های پرکاربرد، به‌خصوص الگوریتم‌های با توانایی تشخیص خوشه‌های با چگالی متفاوت، با استفاده از روش MapReduce پرداخت. پیشنهاد ما موازی‌سازی الگوریتم MD-DBSCAN [۳۷] و الگوریتم LDDBSCAN [۳۰] با استفاده از MapReduce است.

❖ ارائهٔ یک الگوریتم خوشه‌بندی مبتنی بر چگالی قابل‌استفاده در پایگاه داده‌های حجیم با چگالی متفاوت با کمترین وابستگی به پارامترهای ورودی

به‌طور کلی، الگوریتم‌های خوشه‌بندی باید دارای ویژگی‌های خاصی باشند، Andreopoulos این ویژگی‌ها را در [۶۰] چنین توصیف کرده است:

- مقیاس‌پذیری: در زمان کار با مجموعه داده‌های بسیار بزرگ پیچیدگی زمانی نباید در حد انفجار برسد.
- مقاوم بودن: الگوریتم خوشه‌بندی باید قابلیت تشخیص نقاط دورافتاده و نویز را از مجموعه داده داشته باشد.
- عدم حساسیت به ترتیب ورودی: الگوریتم نباید به ترتیب داده‌های ورودی حساس باشد.
- حداقل پارامتر ورودی: تعداد پارامترهایی که توسط کاربر باید تعیین شوند، باید حداقل باشد.
- توانایی خوشه‌بندی اشکال اختیاری.

علاوه بر ویژگی‌های اشاره‌شده در بالا، با توجه به اینکه یکی از ویژگی‌های مهم مجموعه داده‌های دنیای واقعی وجود چگالی‌های مختلف در این مجموعه داده‌هاست، ضروری است که الگوریتم‌های خوشه‌بندی قابلیت تشخیص خوشه‌های با چگالی متفاوت را نیز داشته باشند. در حال حاضر هیچ الگوریتم واحدی که به همهٔ این ویژگی‌ها پرداخته باشد وجود ندارد. ازاین‌رو می‌توانیم تلاش کنیم که اولاً الگوریتم‌هایی برای حل مشکلات موجود ارائه دهیم و در نهایت سعی کنیم تا الگوریتم واحدی ارائه دهیم که تا جای ممکن بتواند همهٔ ویژگی‌های موردنظر را پوشش دهد؛ یعنی ارائهٔ یک الگوریتم خوشه‌بندی مبتنی بر چگالی که:

- توانایی تشخیص خوشه‌های با اشکال، اندازه و چگالی متفاوت را داشته باشد.
- حداقل پیچیدگی زمانی را داشته باشد تا بتوان از آن در مجموعه داده‌های با ابعاد و حجم بالا استفاده کرد.
- در تشخیص خوشه‌های چسبیده‌به‌هم محدودیت نداشته باشد (به ترتیب پردازش اشیاء وابسته نباشد).
- نیاز به حداقل پارامتر ورودی داشته باشد.
- قابلیت خوشه‌بندی اشکال هندسی غیر از نقطه مانند چندضلعی‌ها را داشته باشد.

ارائهٔ چنین الگوریتمی نیاز به آن دارد که ما ابتدا هر مشکل را به‌صورت جداگانه مورد بررسی قرار دهیم و راه‌حلی برای آن ارائه دهیم که این خود منجر به ارائهٔ چندین الگوریتم خوشه‌بندی خواهد شد، مگر آنکه برای حل مشکل موردنظر الگوریتم مناسبی وجود داشته باشد. برای حل هریک از این مشکلات، همان گونه که در این مقاله مشاهده کردید، روش‌های بسیاری ارائه شده است. اما همهٔ این کارها در رابطه با الگوریتم‌های خوشه‌بندی مبتنی بر چگالی بوده که توانایی تشخیص خوشه‌های با چگالی متفاوت را نداشته‌اند و هدف ما انجام چنین بهبودهایی بر روی یک الگوریتم خوشه‌بندی است که توانایی تشخیص خوشه‌های با چگالی متفاوت و پشتیبانی از اشکال هندسی غیر نقطه را داشته باشد. برای نمونه برای بهبود پیچیدگی زمانی الگوریتم مربوطه می‌توان از ایدهٔ به‌کاررفته در روش‌های خوشه‌بندی مبتنی بر مدل استفاده کرد و تعداد پرس‌وجوهای ناحیه‌ای را که در واقع بخش زمان‌بر یک الگوریتم خوشه‌بندی مبتنی بر چگالی هستند، حداقل کرد. یکی از کاربردهای الگوریتم پیشنهادی در خوشه‌بندی تصاویر دریافتی از ماهواره‌هاست.

جدول (۶): ارزیابی انتقادی بهبودهای الگوریتم DBSCAN

ردیف	نام الگوریتم	سال ارائه	ویژگی الگوریتم/ مزایا/ معایب	پارامترها	پیچیدگی زمانی	چند چگالی
۱	DBSCAN [6]	1996	- قابلیت تشخیص خوشه‌های با اشکال و اندازهٔ مختلف، مقاوم در مقابل نویز - عدم تشخیص خوشه‌های با چگالی متفاوت، پیچیدگی زمانی بالا، سخت بودن تعیین مقدار پارامترها، تشخیص ناصحیح خوشه‌های چسبیده‌به‌هم	Eps, Minpts	O(nlogn)	No
۲	GDBSCAN [23]	1998	- به‌منظور کشف خوشه‌های با اشکال هندسی غیر از نقطه	NPred, MinCard, wCard	O(nlogn)	No
۳	OPTICS [9]	1999	- به‌منظور حل مسئلهٔ تغییرات چگالی DBSCAN - تولید خوشه‌های با چگالی محلی بیش از یک حد آستانه به‌جای تولید خوشه‌های با چگالی محلی مشابه	Eps, Minpts	O(nlogn)	Yes
۴	FDBSCAN [24]	2000	- به‌منظور بهبود سرعت DBSCAN - تعداد اشیاء ازدست‌رفته نسبت به DBSCAN بیشتر است	Eps, Minpts	approximatly linear - Much less than that of O(nlogn)	No
۵	PDBSCAN [48]	2002	- به‌منظور افزایش مقیاس‌پذیری و رفع مشکل پیچیدگی زمانی DBSCAN در مواجهه با مجموعه داده‌های بزرگ	Eps, Minpts	Not discussed	No
۶	DBRS [25]	2003	- به‌منظور افزایش مقیاس‌پذیری و کاهش زمان اجرا DBSCAN برای مجموعه داده‌های با چگالی متفاوت - عدم توانایی در ادغام خوشه‌های کوچک	Eps, Minpts, MinPur	approximatly linear	Yes
۷	An enhancement of DBSCAN [44]	2004	- به‌منظور افزایش مقیاس‌پذیری و کاهش زمان اجرا DBSCAN - کارایی بهتر نسبت به الگوریتم DBSCAN و افزایش سرعت تا پنج برابر الگوریتم DBSCAN	Eps, Minpts	Not discussed	No
۸	Parallel-DBSCAN [49]	2006	- به‌منظور افزایش مقیاس‌پذیری و رفع مشکل پیچیدگی زمانی DBSCAN در مواجهه با مجموعه داده‌های بزرگ	Eps, Minpts	Not discussed	No
۹	VDBSCAN [26]	2007	- به‌منظور حل مسئله تغییرات چگالی DBSCAN - تنها برای برخی مجموعه داده‌های خاص که دارای منحنی ملایم در k-dist Plot متناظرشان نیستند، صحیح کار می‌کند	k	O(nlogn)	Yes
۱۰	ST-DBSCAN [29]	2007	- به‌منظور کشف خوشه‌ها مطابق با مقادیر مکانی، غیرمکانی و زمانی اشیاء - عدم پشتیبانی از خوشه‌های با چگالی متفاوت - عدم تعیین خودکار پارامترهای ورودی	Eps1, Eps2, Minpts, $\Delta\varepsilon$	O(nlogn)	No
۱۱	LDBSCAN [30]	2007	- به‌منظور حل مسئله تغییرات چگالی DBSCAN - مشکل بودن تعیین مقدار پارامترها به‌دلیل تعداد زیاد پارامترهای ورودی	LofUB, Pct, Minpts	- O(n) For Low-Dimensional Data - O(nlogn) For medium-Dimensional Data - $O(n^2)$ For High-Dimensional	Yes

ردیف	نام الگوریتم	سال ارائه	ویژگی الگوریتم/ مزایا/ معایب	پارامترها	پیچیدگی زمانی	چند چگالی
					Data	
۱۲	DDSC [32]	2008	- به‌منظور حل مسئله تغییرات چگالی DBSCAN - کاهش وابستگی به پارامتر Eps	Eps, Minpts, α	O(nlogn)	Yes
۱۳	GF-DBSCAN [33]	2009	- به‌منظور بهبود پیچیدگی زمانی DBSCAN - سرعت بالاتر نسبت به الگوریتم FDBSCAN - قابلیت تولید خوشه‌های پایدار	Eps, Minpts	approximatly linear	No
۱۴	Incremental DBSCAN [39]	2009	- یک نسخه افزایشی از الگوریتم DBSCAN برای استفاده در محیط‌های انباره داده - زمانی که میزان نویز پایین باشد، این الگوریتم نسبت به سایر الگوریتم‌های افزایشی موجود مقیاس‌پذیرتر بوده و نیاز به پرس‌وجوی ناحیه‌ای کمتری دارد - عدم توانایی حذف افزایشی خوشه‌ها - عدم پشتیبانی از خوشه‌های با چگالی متفاوت	Eps, Minpts	Not discussed	No
۱۵	ODBSCAN [34]	2010	- به‌منظور بهبود سرعت الگوریتم DBSCAN - دارای پیچیدگی زمانی کمتر از DBSCAN و FDBSCAN - دارای دقت بالاتر نسبت به الگوریتم FDBSCAN	Eps, Minpts	approximatly linear	No
۱۶	DVBSCAN [35]	2010	- به‌منظور حل مسئلهٔ تغییرات چگالی DBSCAN - مشکل‌تر بودن تعیین مقدار پارامترها به‌دلیل تعداد بالای پارامترها - پیچیدگی زمانی بالا	Eps, Minpts, α, γ	Not discussed	Yes
۱۷	IVDBSCAN [36]	2010	- به‌منظور کمک به انتخاب خودکار پارامترهای VDBSCAN - دقت و کارایی بالاتر نسبت به الگوریتم VDBSCAN - سرعت بالاتر نسبت به الگوریتم VDBSCAN	Computed Automatically	Less than that of O(nlogn)	Yes
۱۸	PDBSCAN With PR-Tree [50]	2010	- به‌منظور افزایش مقیاس‌پذیری و رفع مشکل پیچیدگی زمانی DBSCAN در مواجهه با مجموعه داده‌های بزرگ	Eps, Minpts	Not discussed	No
۱۹	KDDClus [28]	2011	- به‌منظور حل مسئلهٔ تغییرات چگالی DBSCAN - دارای قابلیت پردازش کارای داده‌های با ابعاد بالا	k	O(nlogn)	Yes
۲۰	MD-DBSCAN [37]	2011	- به‌منظور حل مسئله تغییرات چگالی DBSCAN - انتخاب مقدار نامناسب برای پارامتر k منجر به مسائل پل نویز و شکاف خوشه‌ها می‌شود.	K, Var	Not discussed	Yes
۲۱	VMDBSCAN [38]	2011	- به‌منظور حل مسئله تغییرات چگالی DBSCAN - نسبت به الگوریتم DBSCAN تعداد خوشه‌های صحیح‌تری را تشخیص می‌دهد. - برای داده‌های با بعد بالا تعداد خوشه‌های ناصحیح قابل‌توجه است.	Eps, Minpts, η	More than that of O(nlogn)	Yes
۲۲	VDSC [40]	2011	- به‌منظور حل مسئله تغییرات چگالی DBSCAN - دارای کارایی بالاتر نسبت به الگوریتم DVBSCAN - قابلیت تشخیص خوشه‌های تودرتو در محیط با چگالی متفاوت	Minpts, α	O(nlogn)	Yes
۲۳	Improved DBSCAN [41]	2012	- به‌منظور افزایش مقیاس‌پذیری DBSCAN و تشخیص خوشه‌های با چگالی متفاوت - تعداد خوشه‌های صحیح یافت‌شده در این الگوریتم نسبت	Minpts, Eps	Not discussed	Yes

ردیف	نام الگوریتم	سال ارائه	ویژگی الگوریتم / مزایا / معایب	پارامترها	پیچیدگی زمانی	چند چگالی
			به DBSCAN بیشتر است - امکان از دست رفتن برخی نقاط مرکزی			
۲۴	AutoEps DBSCAN [27]	2013	- به‌منظور حل مسئله تغییرات چگالی DBSCAN و کاهش وابستگی به پارامترهای ورودی - عملکرد مناسب در مواجهه با مجموعه داده‌های بزرگ - کارایی بهتر نسبت به الگوریتم VDBSCAN	K	O(nlogn)	Yes
۲۵	Revised DBSCAN [42]	2013	- به‌منظور رفع مشکل تشخیص ناصحیح خوشه‌های مجاور - قابلیت استفاده بر روی داده مکانی و غیرمکانی - مستقل بودن نتایج خوشه‌بندی از ترتیب کشف خوشه‌ها	Eps, Minpts	Not discussed	No
۲۶	Improvement of DBSCAN [43]	2013	- به‌منظور افزایش مقیاس‌پذیری الگوریتم DBSCAN - با توجه به اینکه این الگوریتم تعداد صحیح‌تری از خوشه‌ها را نسبت به DBSCAN کشف می‌کند از دقت بالاتری نسبت به DBSCAN برخوردار است.	Eps, Minpts	Not discussed	No
۲۷	O-VDBSCAN [47]	2013	- به‌منظور تعیین خودکار پارامتر k الگوریتم VDBSCAN - پیچیدگی زمانی بالا به‌دلیل بزرگ بودن دامنه اولیه k	Computed Automatically	More than that of O(nlogn)	Yes

مراجع

[1] Güting, Ralf Hartmut, "*An introduction to spatial database systems*", The VLDB Journal—The International Journal on Very Large Data Bases, vol. 3, no. 4, pp. 357-399, Oct. 1994.

[2] J. Mennis and D. Guo, "*Spatial data mining and geographic knowledge discovery—An introduction*", Computers, Environment and Urban Systems, vol. 33, no. 6, pp. 403-408, Nov. 2009.

[3] C. J. Matheus, P. K. Chan, and G. Piatetsky-Shapiro, "*Systems for knowledge discovery in databases*", IEEE Trans. on Knowledge and Data Engineering, vol. 5, no. 6, pp. 903-913, Dec. 1993.

[4] N. Soni and A. Ganatra, "*Comparative study of several Clustering Algorithms*", International Journal of Advanced Computer Research (IJACR), vol. 2, no. 4, pp. 37-42, Dec. 2012.

[5] N. Soni and A. Ganatra, "*Categorization of Several Clustering Algorithms from Different Perspective: A Review*", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 2, no. 8, pp. 63-68, Aug. 2012.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "*A density-based algorithm for discovering clusters in large spatial databases with noise*", in Kdd, vol. 96, no. 34, pp. 226-231, Aug. 1996.

[7] Hinneburg, Alexander, and Daniel A. Keim, "*An efficient approach to clustering in large multimedia databases with noise*", in KDD, vol. 98, pp. 58-65, Aug. 1998.

[8] Xu, Xiaowei, et al, "*A distribution-based clustering algorithm for mining in large spatial databases*", in Proc. of 14th Int. Conf. on Data Engineering (ICDE'98), pp. 324-331, Feb. 1998.

[9] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "*Optics: Ordering points to identify the clustering structure*", in ACM Sigmod Record, vol. 28, no. 2, pp. 49-60, Jun. 1999.

[10] He, Ling, Ling-da WU, and Yi-chao CAI, "*Survey of Clustering Algorithms in Data Mining [J]*", Application research of computers, vol. 1, no. 1, pp. 10-13, Jun. 2007.

[11] Xu, Rui, and Donald Wunsch, "*Survey of clustering algorithms*", IEEE Trans. on neural networks, vol. 16, no. 3, pp. 645-678, May. 2005.

[12] P. Berkhin, "*A survey of clustering data mining techniques*", Grouping multidimensional data, vol. 25, pp. 25-71, Feb. 2006.

[13] Parimala, M., Daphne Lopez, and N. C. Senthilkumar, "*A survey on density based clustering algorithms for mining large spatial databases*",

International Journal of Advanced Science and Technology , vol. 31, no. 1, pp. 59-66, Jun. 2011.

[14] T. Liu, C. Rosenberg, and H. A. Rowley, "*Clustering billions of images with large scale nearest neighbor search*", in Proc. IEEE Workshop on Applications of Computer Vision, Vol. 7, pp. 28-28, Feb. 2007.

[15] Oyelade, O. J., O. O. Oladipupo, and I. C. Obagbuwa, "*Application of k Means Clustering algorithm for prediction of Students Academic Performance*", International Journal of Computer Science and Information Security(IJCSIS), vol. 7, no. 1, pp. 292-295, Feb. 2010.

[16] Akkaya, Kemal, Fatih Senel, and Brian McLaughlan, "*Clustering of wireless sensor and actor networks based on sensor distribution and connectivity*", Journal of Parallel and Distributed Computing, vol. 69, no.6, pp. 573-587, Jun. 2009.

[۱۷] یوسفان، احمد، یوسفیان، الهام، «**خوشه‌بندی استان‌های ایران بر پایه‌ی معیارهای شکاف دیجیتال به‌کمک روش K-MEANS**»، نشریه علمی‌ترویجی محاسبات نرم، ۱ (۱)، ۴۵-۳۲، ۱۳۹۱.

[18] J. MacQueen, "*Some methods for classification and analysis of multivariate observations*", in *Proc. of the Fifth Berkeley Symp. On Mathematical Statistics and Probability*, vol. 1, no. 14, pp. 281-297, Jun. 1967.

[۱۹] صادق‌زاده، نگار، شمسی، محبوبه، رسولی کناری، عبدالرضا. «**حاشیه‌نویسی تصویر با استفاده از الگوریتم خوشه‌بندی نیمه نظارتی طیفی**»، نشریه علمی‌ترویجی محاسبات نرم، ۳ (۱)، ۳۵-۲۰، ۱۳۹۳.

[20] R. O. Duda, P. E. Hart, D. G. Stork, "*Pattern Classification And Scene Analysis*", John Wiley & Sons, New York, Nov. 2000.

[21] M. Celik, F. Dadaser-Celik, and A. Dokuz, "*Anomaly detection in temperature data using DBSCAN algorithm*", in Proc. Int. Symp. On Innovations in Intelligent Systems and Applications, vol. 1, pp. 91-95, Istanbul, Turkey, Jun. 2011.

[22] N. Maitry, D. Vaghela, "*Survey on Different Density Based Algorithms on Spatial Dataset*", International Journal of Advance Research in Computer Science and Management Studies, vol. 2, no. 2, pp. 362-366, Feb. 2014.

[23] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "*Density-based clustering in spatial databases: The algorithm gdbscan and its applications*", Data mining and knowledge discovery, vol. 2, no. 2, pp. 169-194, Jun. 1998.

[24] S. Zhou, A. Zhou, W. Jin, Y. Fan, and W. Qian, "*FDBSCAN: a fast DBSCAN algorithm*", Journal of Software, vol. 11, no. 6, pp. 735-744, Jun. 2000.

[25] X. Wang and H. J. Hamilton, "*DBRS: a density-based spatial clustering method with random sampling*", in Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining, PAKDD'03, pp. 563-575, Apr. 2003.

[26] P. Liu, D. Zhou, and N. Wu, "*VDBSCAN: varied density based spatial clustering of applications with noise*", in Proc. Int. Conf. on Service Systems and Service Management, pp. 1-4, Chengdu, China, Jun. 2007.

[27] M. N. Gaonkar and K. Sawant, "*AutoEpsDBSCAN: DBSCAN with Eps automatic for large dataset*", International Journal on Advanced Computer Theory and Engineering, vol. 2, no. 2, pp. 2319-2526, Aug. 2013.

[28] S. Mitra and J. Nandy, "*KDDClus: A Simple Method for Multi-Density Clustering*", *in* Proc. of the Int. Workshop on Soft Computing Applications and Knowledge Discovery, SKAD'11, vol. 13, pp. 72-76, Moscow, Russia, Jun. 2011.

[29] D. Birant and A. Kut, "*ST-DBSCAN: An algorithm for clustering spatial–temporal data*", Data & Knowledge Engineering, vol. 60, no. 1, pp. 208-221, Jan. 2007.

[30] L. Duan, L. Xu, F. Guo, J. Lee, and B. Yan, "*A local-density based spatial clustering algorithm with noise*", Information Systems, vol. 32, no. 7, pp. 978-986, Nov. 2007.

[31] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "*LOF: identifying density-based local outliers*", *in* Proc. of the 1999 ACM Int. Conf. on Management of data, SIGMOD'99, vol. 29, no. 2, pp. 93-104, May. 2000.

[32] B. Borah and D. K. Bhattacharyya, "*DDSC: a density differentiated spatial clustering technique*", Journal of Computers, vol. 3, no. 2, pp. 72-79, Feb. 2008.

[33] C.-F. Tsai, C.-T. Wu, and S. Chen, "GF-DBSCAN; A New Efficient and Effective Data Clustering Technique for Large Databases", in *Proc. WSEAS Int. Conf. Mathematics and Computers in Science and Engineering*, vol. 9, pp. 231-236, Hangzhou, China, May. 2009.

[34] J. H. Peter and A. Antonysamy, "*An Optimised Density Based Clustering Algorithm*", International

Journal of Computer Applications, vol. 6, no. 9, pp. 20-25, Sep. 2010.

[35] A. Ram, S. Jalal, A. S. Jalal, and M. Kumar, "*A density based algorithm for discovering density varied clusters in large spatial databases*", International Journal of Computer Applications, vol. 3, no. 6, pp. 1-4, Jun. 2010.

[36] S. Vijayalakshmi and M. Punithavalli, "*Improved varied density based spatial clustering algorithm with noise*", in Proc. Int. Conf. on Computational Intelligence and Computing Research (ICCIC), pp. 1-4, Dec. 2010.

[37] W. Ashour and S. Sunoallah, "*Multi density DBSCAN*", in Intelligent Data Engineering and Automated Learning-IDEAL, Ed: Springer, pp. 446-453, Sep. 2011.

[38] M. T. Elbatta, R. M. Bolbol, and W. M. Ashour, "*A vibration method for discovering density varied clusters*", International Scholarly Research Network, vol. 2012, Article ID 723516, pp. 1-8, Nov. 2011.

[39] Goyal, Navneet, et al, "*An efficient density based incremental clustering algorithm in data warehousing environment*", in Int. Conf. on Computer Engineering and Applications, vol. 2, pp. 482-486, Aug. 2011.

[40] R. K. Prasad and R. Sarmah, "Variable density spatial data clustering", in *Proc. 2nd Int. Conf. on Computer and Communication Technology, ICCCT'11*, vol. 2, pp. 53-58, Jun. 2011.

[41] Shah, Glory H. "An improved DBSCAN, a density based clustering algorithm with parameter selection for high dimensional data sets", in Proc. IEEE 2012 Nirma University Int. Conf. on Engineering, pp. 1-6, Dec. 2012.

[42] T. N. Tran, K. Drab, and M. Daszykowski, "*Revised DBSCAN algorithm to cluster data with dense adjacent clusters*", Chemometrics and Intelligent Laboratory Systems, vol. 120, no. 1, pp. 92-96, Jan. 2013.

[43] Dharni, Chetan, and Meenakshi Bnasal, "*An improvement of DBSCAN Algorithm to analyze cluster for large datasets*", in Proc. IEEE int. conf. in MOOC, Innovation and Technology in Education (MITE), pp. 42-46, Dec. 2013.

[44] Y. El-Sonbaty, M. Ismail, and M. Farouk, "*An efficient density based clustering algorithm for large databases*", in Proc. 16th IEEE int. Conf. on Tools with Artificial Intelligence(ICTAI), pp. 673-677, Nov. 2004.

[45] R. T. Ng and J. Han, "*Clarans: A method for clustering objects for spatial data mining*", IEEE Trans. on Knowledge and Data Engineering, vol. 14, no. 5, pp. 1003-1016, Sep. 2002.

[46] G. Karypis, E.-H. Han, and V. Kumar, "*Chameleon: Hierarchical clustering using dynamic modeling*", Computer, vol. 32, no. 8, pp. 68-75, Aug. 1999.

[47] Wang, Wei, Shuang Zhou, and Qingqing Xiao. "*Optimum Vdbscan (O-VDBSCAN) For Identifying Downtown Areas*", International Journal of Digital Information and Wireless Communications (IJDIWC), vol. 3, no. 3, pp. 271-276, Dec. 2013.

[48] Xu, Xiaowei, Jochen Jäger, and Hans-Peter Kriegel, "*A fast parallel clustering algorithm for large spatial databases*", in High Performance Data Mining, vol. 3, pp. 263-290, Jan. 2002.

[49] Brecheisen, Stefan, Hans-Peter Kriegel, and Martin Pfeifle, "*Parallel density-based clustering of complex objects*", in Proc. Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining, PAKDD'06, vol. 3918, pp. 179-188, Apr. 2006.

[50] Chen, Min, Xuedong Gao, and HuiFei Li, "*Parallel DBSCAN with priority r-tree*", in Proc. IEEE Int. Conf. on Information Management and Engineering (ICIME), pp. 508-811, Apr. 2010.

[51] Berchtold, Stefan, Daniel A. Keim, and Hans-Peter Kriegel, "*The X-tree: An index structure for high-dimensional data*", in Proc. of the 22th Int. Conf. on Very Large DataBases, vol. 96, pp. 28-39, Sep. 1996.

[52] Kamel I., Faloutsos C., "*Hilbert R-tree: An Improved R-tree using Fractals*", In Proc. of the 20th Int. Conf. on Very Large Databases, pp. 500-509, Feb. 1993.

[53] '*Clustering datasets*'. [Online]. Available: http://cs.joensuu.fi/sipu/datasets/. [Accessed: 6-Aug-2016].

[54] W. M. Rand, "*Objective criteria for the evaluation of clustering methods*", Journal of the American Statistical association, vol. 66, no. 336, pp. 846-850, Dec. 1971.

[55] P. Jaccard, "*Distribution of the alpine flora in the dranse's basin and some neighbouring regions*", Bulletin de la Societe vaudoise des Sciences Naturelles, vol. 37, no. 1, pp. 241-272, Jan. 1901.

[56] E. B. Fowlkes and C. L. Mallows, "*A method for comparing two hierarchical clusterings*", Journal of the American Statistical association, vol. 78, no. 383, pp. 553-569, Sep. 1983.

[57] Kriegel, Hans-Peter, Peer Kröoger, and Irina Gotlibovich, "*Incremental OPTICS: Efficient computation of updates in a hierarchical cluster*

ordering", in Proc. of the 5th Int. Conf. on Data Warehousing and Knowledge Discovery, vol. 2737, pp. 224-233, Sep. 2003.

[58] Dean, J., and Ghemawat, S., "*MapReduce: simplified data processing on large clusters*", Communications of the ACM, vol. 51, no. 1, pp. 107-113, Jan. 2008.

[59] He, Yaobin, Haoyu Tan, Wuman Luo, Huajian Mao, Di Ma, Shengzhong Feng, and Jianping Fan, "*Mr-dbscan: An efficient parallel density-based clustering algorithm using mapreduce*", in Proc. IEEE 17th Int. Conf. on Parallel and Distributed Systems (ICPADS), pp. 473-480, Dec. 2011.

[60] B. Andreopoulos, A. An, X. Wang, and M. Schroeder, "*A roadmap of clustering algorithms: finding a match for a biomedical application*", Briefings in Bioinformatics, vol. 10, no. 3, pp. 297-314, Feb. 2009.