

VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING



Assignment Khai phá dữ liệu

Streaming/ Big data clustering

Giảng viên hướng dẫn: Lê Hồng Trang

Sinh viên thực hiện:

Trần Quang Khả - 1913764

Nguyễn Chí Tuệ - 1915795

Võ Thị Na - 1914210

Trương Công Hoàng - 1913459

Lê Hồng Nhật - 1914474



Mục lục

1	Mở đầu	3
2	Variants K-means	3
2.1	Giới thiệu	3
2.2	Thuật toán	3
2.2.1	Giải thuật K-means	3
2.2.2	Giải thuật K-means-plus-plus	4
2.2.3	Giải thuật Minibatch K-means	4
2.3	Ứng dụng	5
2.3.1	Ứng dụng trong phát hiện bất thường	5
2.3.2	Một số ứng dụng khác	6
2.4	Thực nghiệm và đánh giá	6
2.4.1	HAWKS-generated datasets	6
2.4.1.a	K-means	7
2.4.1.b	K-means-plus-plus	7
2.4.1.c	Minibatch K-means	8
2.4.1.d	Bảng so sánh thời gian chạy của ba thuật toán	8
2.4.2	Thử nghiệm với tập dữ liệu Birch 1	8
3	BIRCH	10
3.1	Giới thiệu	10
3.2	Phương pháp TOP DOWN	11
3.3	Thuật toán	11
3.4	Ứng dụng	11
3.5	Thực nghiệm và đánh giá	12
3.5.1	Thử nghiệm với tập dữ liệu Birch 1	12
3.5.2	Thử nghiệm với tập dữ liệu Birch 2	14
3.5.3	Thử nghiệm với tập dữ liệu Birch 3	14
3.5.4	Nhận xét và đánh giá	16
4	CLARANS	16
4.1	Giới thiệu	16
4.1.1	Giải thuật PAM	16
4.1.2	Giải thuật CLARA	17
4.1.3	Giải thuật CLARANS	17
4.2	Hiện thực	18
4.3	Ứng dụng	19
4.4	Thực hiện và đánh giá	19



4.4.1	Thử nghiệm với tập dữ liệu sinh ra bởi HAWKS	19
4.4.2	Thử nghiệm với tập dữ liệu Birch	22
4.4.2.a	Thử nghiệm với tập dữ liệu Birch1	25
4.4.2.b	Thử nghiệm với tập dữ liệu Birch2	27
4.4.2.c	Thử nghiệm với tập dữ liệu Birch3	28
4.4.2.d	Nhận xét	29
5	DBSCAN	30
5.1	Giới thiệu	30
5.2	Hiện thực	31
5.3	Ứng dụng	32
5.4	Thực nghiệm và đánh giá	32
5.4.1	Thử nghiệm với tập dữ liệu sinh ra bởi HAWKS	32
5.4.2	Thử nghiệm với tập dữ liệu Birch1	34
5.5	Thử nghiệm với tập dữ liệu Birch2	36
5.6	Thử nghiệm với tập dữ liệu Birch3	37
6	Tài liệu tham khảo	39



1 Mở đầu

2 Variants K-means

2.1 Giới thiệu

K-means là một thuật toán phân cụm đơn giản thuộc loại dữ liệu không có nhãn và được sử dụng để giải quyết bài toán phân cụm. Tương tự của thuật toán phân cụm k-means là phân chia 1 bộ dữ liệu thành các cụm khác nhau. Trong đó số lượng cụm được cho trước là k .

Công việc phân cụm được xác lập dựa trên nguyên lý: Các điểm dữ liệu trong cùng 1 cụm thì phải có cùng 1 số tính chất nhất định. Tức là giữa các điểm của cụm 1 cụm phải có sự liên quan lẫn nhau. Thuật toán phân cụm được sử dụng trong các ứng dụng cõi máy tìm kiếm, phân đoạn khách hàng, thống kê dữ liệu,..

Tuy nhiên, thuật toán K-means còn một số hạn chế như:

- Hạn chế về mặt thời gian
- Các cụm cần có số lượng điểm dữ liệu gần bằng nhau và hình dạng của các cụm phải hình dạng lồi
- Chúng ta cần phải biết trước số lượng cụm cần phải gom cụm. Số k của thuật toán phải được chọn trước. Nhưng trong thực tế có nhiều trường hợp chúng ta phải phân cụm mà không biết được số lượng của chúng
- Kết quả cuối cùng của việc phân cụm phụ thuộc nhiều vào cái centers được khởi tạo ban đầu. Do vậy nếu vị trí ban đầu được khởi tạo ngẫu nhiên không tốt thì kết quả của phân cụm cũng không được tốt

2.2 Thuật toán

2.2.1 Giải thuật K-means

Mã giả của giải thuật Kmean như sau:

1. Chọn K điểm bất kỳ làm các *center* ban đầu (có thể là K điểm của tập dữ liệu). Mỗi *center* sẽ là điểm trung tâm của một cụm.
2. Với mỗi điểm dữ liệu, tính toán khoảng cách của nó đến *center* và gán cho nó vào cụm có *center* gần với điểm dữ liệu này nhất.
3. Nếu ở bước 2 không có điểm dữ liệu nào bị thay đổi cụm hoặc gán vào cụm mới thì ta dừng thuật toán



4. Tính trung bình cộng của tất cả các dữ liệu trong một cụm, sau đó gán trung bình cộng tính được vào các *center*.
5. Quay lại bước 2.

Độ phức tạp của mỗi vòng lặp là **kn** với k là số *center* và n là số điểm dữ liệu.

2.2.2 Giải thuật K-means-plus-plus

Với giải thuật K-means thông thường, các *center* ban đầu được chọn ngẫu nhiên. Tuy nhiên việc chọn ngẫu nhiên dẫn đến kết quả phân cụm không được tốt. Giải thuật K-means-plus-plus là một phương pháp khác cho việc lựa chọn các điểm dữ liệu ban đầu.

Phương pháp đó được trình bày như sau:

1. Chọn một điểm dữ liệu ngẫu nhiên từ các điểm dữ liệu làm *center* thứ nhất.
2. Đối với mỗi điểm dữ liệu còn lại, tính khoảng cách của nó với các *center* đã được chọn và chọn ra trung tâm có khoảng cách ngắn nhất.
3. Chọn điểm kế tiếp trong tập dữ liệu làm trung tâm, sao cho xác suất để một điểm được chọn tỉ lệ thuận với khoảng cách của nó đến *center* đã chọn gần nó nhất
4. Nếu chưa chọn được K *center* thì quay lại bước 2. Ngược lại dừng thuật toán.

Nhận xét: Từ bước 3, các *center* được chọn sẽ có khoảng cách xa nhau. Do đó khả năng các *center* được chọn sẽ ở các cụm khác nhau của tập dữ liệu sẽ tăng lên.

2.2.3 Giải thuật Minibatch K-means

Đối với giải thuật K-means sau mỗi vòng lặp tất cả các điểm dữ liệu đều phải được lưu trữ trong bộ nhớ chính và phải được tính toán để tìm các *center* mới. Khi số lượng các điểm dữ liệu tăng lên thì thời gian thực hiện giải thuật cũng tăng theo. Giải thuật Minibatch K-means đưa ra để làm giảm thiểu độ phức tạp này.

Ý tưởng chính của giải thuật *Minibatch K-means* là chỉ sử dụng một tập hợp nhỏ ngẫu nhiên các điểm dữ liệu thay vì toàn bộ các điểm dữ liệu sau mỗi vòng lặp tính toán. Sau mỗi vòng lặp, một tập hợp ngẫu nhiên khác sẽ được chọn, mỗi điểm dữ liệu trong tập hợp được chọn ấy sẽ cập nhật *center* của mỗi cụm. Giải thuật cũng áp dụng một learning rate giảm dần sau mỗi vòng lặp. Do đó khi số vòng lặp tăng lên, sự thay đổi của các *center* sẽ giảm dần, đến khi sự thay đổi ấy đủ nhỏ thì ta có thể dừng thuật toán.

Giải thuật *Minibatch K-means* được trình bày dưới đây.



Cho một tập dữ liệu $D = \{d_1, d_2, d_3, \dots, d_n\}$. Số lượng iteration t, kích thước tập con b và số cụm k. Tập các điểm center là $K = \{c_1, c_2, c_3, \dots, c_n\}$

1. Khởi tạo các center c_i ($1 \leq i \leq k$) ban đầu (dùng thuật toán k-means-plus-plus hoặc ngẫu nhiên)
2. Khởi tạo số điểm dữ liệu của mỗi cụm $N_i = 0, 1 \leq i \leq k$
3. Lặp các bước sau đây t lần:
 - (a) Chọn ngẫu nhiên b điểm dữ liệu trong D.
 - (b) Tìm và lưu lại các center gần nhất đối với mỗi điểm dữ liệu.
 - (c) Đối với mỗi điểm dữ liệu d_j :
 - i. Lấy ra center gần nhất, giả sử center đó là c_i
 - ii. Tăng số điểm dữ liệu của cụm N_i lên 1
 - iii. Tính learning rate $lr = 1/N_i$
 - iv. Cập nhật lại center $c_i = (1 - lr) * C_i + lr * d_j$

Nhận xét: Giải thuật *Minibatch K-means* hội tụ nhờ vào sự giảm dần của learning rate sau mỗi vòng lặp. Tiêu chí để dừng thuật toán có thể dựa vào số lượng vòng lặp cho trước hoặc dựa vào sự thay đổi quá nhỏ của các center sau mỗi vòng lặp.

2.3 Ứng dụng

K-means là một thuật toán rất đơn giản nhưng có rất nhiều ứng dụng trong thực tiễn. Một số ứng dụng của thuật toán này có thể kể đến như:

2.3.1 Ứng dụng trong phát hiện bất thường

Thuật toán K-means không thích hợp trong ứng dụng phát hiện bất thường. Do trong thuật toán K-means, tất cả các điểm dữ liệu, kể cả những điểm dữ liệu nhiễu đều được gán vào một cụm nào đó, do đó không thể phân biệt đâu là dữ liệu bất thường đâu là dữ liệu bình thường.

Có thể sử dụng một số kinh nghiệm như:

- Dựa vào số lượng các điểm trong cụm, nếu cụm có quá ít điểm dữ liệu thì có thể cụm đó là bất thường
- Dựa vào khoảng cách các điểm dữ liệu đến center, nếu khoảng cách này lớn hơn một ngưỡng nhất định thì điểm dữ liệu được xem là bất thường.

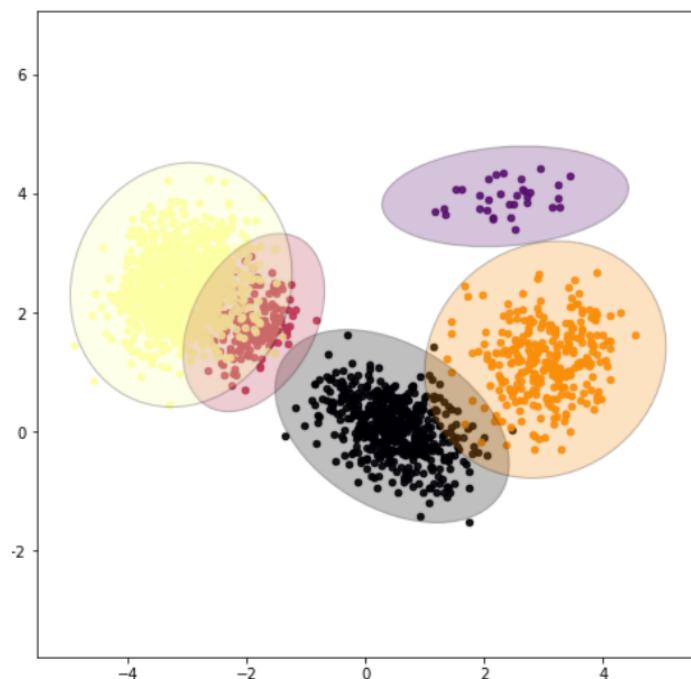
2.3.2 Một số ứng dụng khác

- Phân khúc khách hàng(customer segmentation) trong kinh doanh: dựa vào các hành vi mua hàng, quan tâm,.. để phân loại và đưa ra các chiến dịch kinh doanh hợp ý
- Phân tích gen trong y khoa
- Nén hình ảnh
- Phát hiện tế bào ung thư
- Phân loại văn bản(document classification): Phân cụm các tài liệu dựa chủ đề hoặc thẻ.

2.4 Thực nghiệm và đánh giá

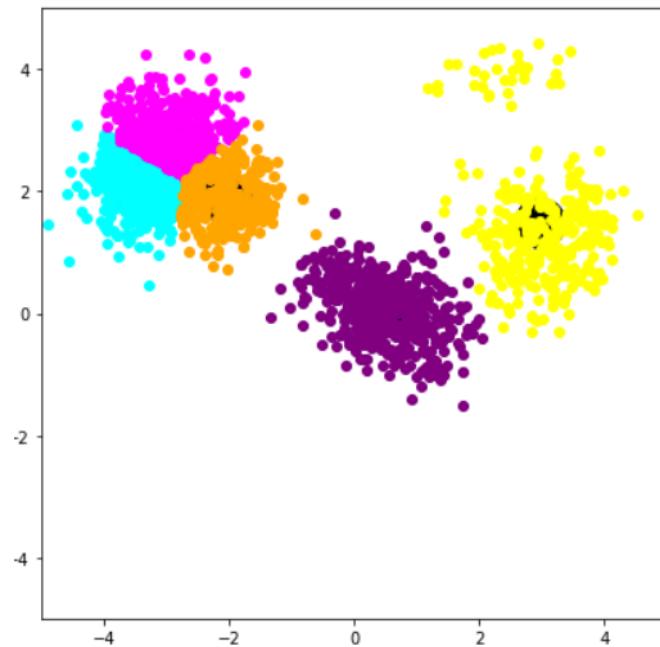
2.4.1 HAWKS-generated datasets

KAWKS generator là một công cụ có khả năng sinh ra các tập dữ liệu nhân tạo cho việc phân cụm dựa vào thuật toán di truyền.Sau đây là tập dữ liệu sinh bởi HAWKS



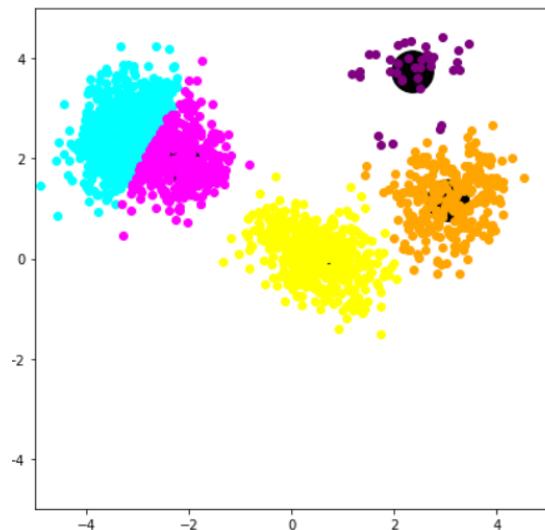
Hình 1: Tập dữ liệu sinh bởi HAWKS

2.4.1.a K-means



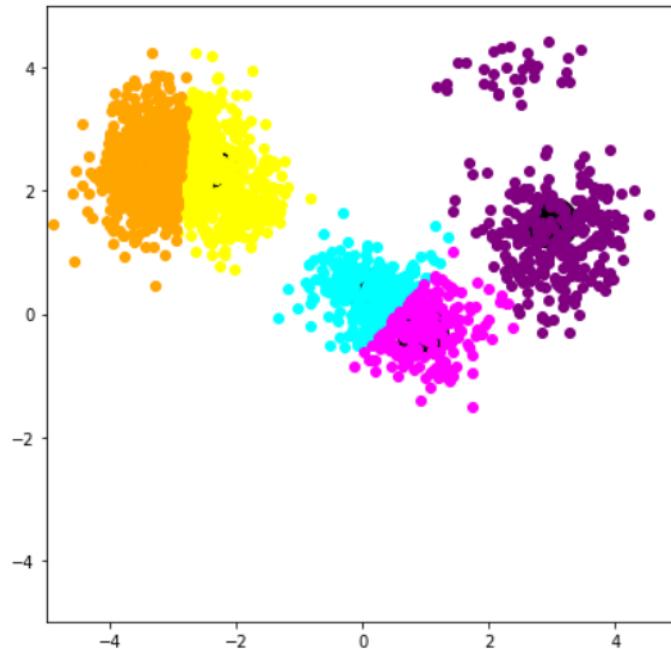
Hình 2: Kết quả phân cụm của Kmeans với tập dữ liệu sinh bởi HAWKS

2.4.1.b K-means-plus-plus



Hình 3: Kết quả phân cụm của K-means-plus-plus với tập dữ liệu sinh bởi HAWKS

2.4.1.c Minibatch K-means



Hình 4: Kết quả phân cụm của Minibatchh-K-means với tập dữ liệu sinh bởi HAWKS

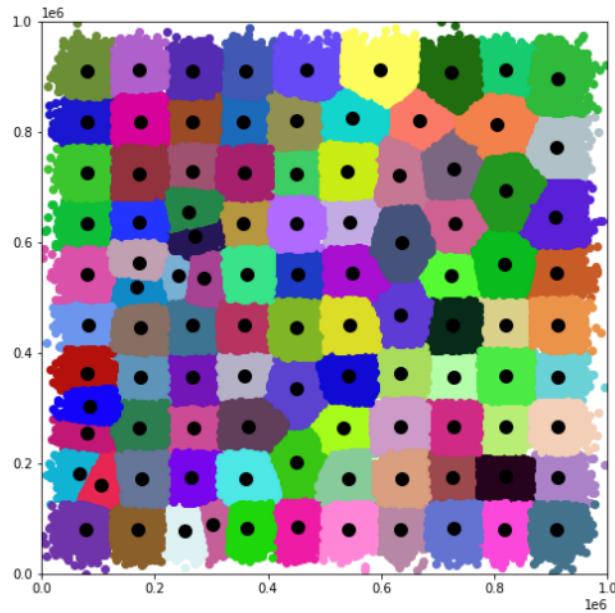
2.4.1.d Bảng so sánh thời gian chạy của ba thuật toán

Kmeans	Kmeans-plus-plus	Minibatch-K-means
938 ms	631 ms	525ms

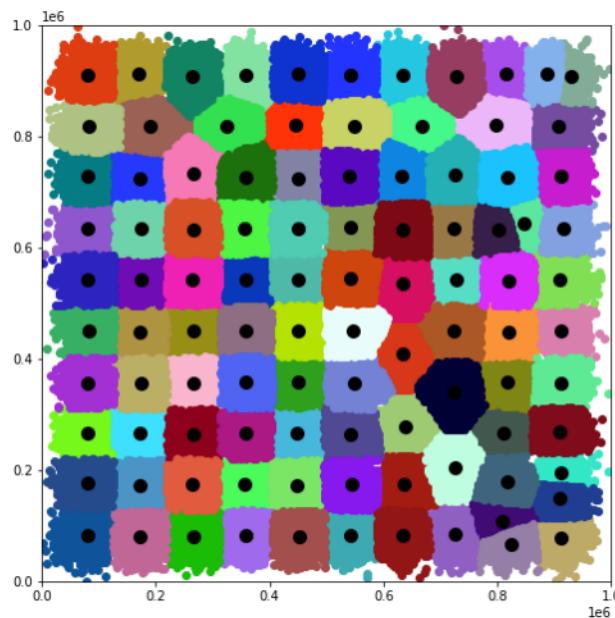
Nhận xét: Cả ba thuật toán đều cho ra kết quả không ổn định đối với tập dữ liệu. Thuật toán K-means-plus-plus thường cho kết quả phân cụm chính xác hơn. Thuật toán Minibatch-K-means hy sinh một ít độ chính xác nhưng cải thiện về mặt thời gian hơn so với thuật toán K-means.

2.4.2 Thử nghiệm với tập dữ liệu Birch 1

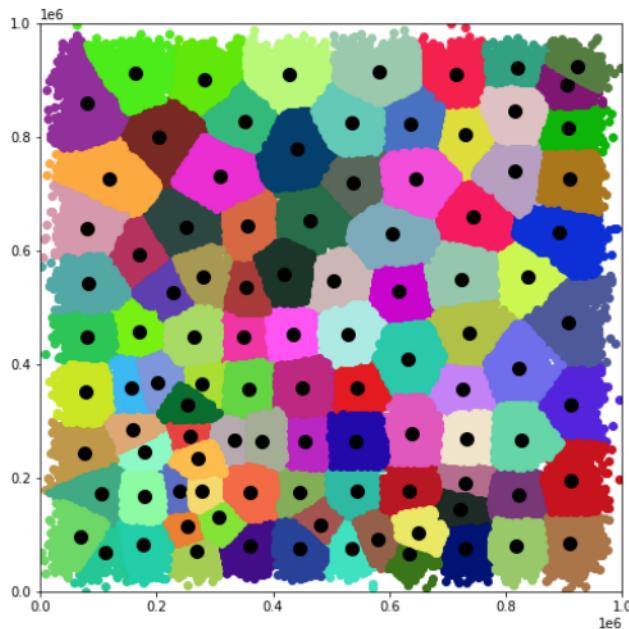
Tiếp tục tiến hành thí nghiệm với tập dữ liệu birch 1, kết quả thí nghiệm như sau.



Hình 5: Birch1 với Kmeans



Hình 6: Birch1 với Kmeans-plus-plus



Hình 7: Birch1 với minibatch Kmeans

So sánh thời gian chạy:

Kmeans	Kmeans-plus-plus	Minibatch-Kmeans
2 min 33 s	1 min 47 s	26.8 s

Nhận xét: Thuật toán K-means-plus-plus có khả năng làm giảm thời gian chạy và tăng độ chính xác của cả 2 thuật toán Kmeans và Minibatch Kmeans. Bên cạnh đó thuật toán Minibatch Kmeans cũng làm giảm thời gian chạy đáng kể, tuy nhiên độ chính xác không được tốt so với thuật toán Kmeans.

3 BIRCH

3.1 Giới thiệu

Birch : Balanced Iterative Reducing Clustering Using Hierarchies được đề xuất năm 1996 bởi Tian Zhang amakrisnan và Livny. Birch là thuật toán phân cụm sử dụng chiến lược TOP DOWN.

Ưu điểm của Birch là nó có khả năng gom cụm tăng dần và động các điểm dữ liệu đa chiều, cố gắng phân cụm hiệu quả nhất dựa trên điều kiện tự nhiên hiện có (bộ nhớ, thời gian thực thi). Trong hầu hết các trường hợp, Birch chỉ cần quét một lần qua toàn bộ cơ sở dữ liệu



3.2 Phương pháp TOP DOWN

Phương pháp Top Down : Bắt đầu với trạng thái tất cả các đối tượng được sắp xếp trong cùng một cụm.

Mỗi vòng lặp thành công, một cụm được tách thành các cụm nhỏ hơn theo giá trị của phép đo độ tương tự nào đó cho đến khi mỗi đối tượng là một cụm hoặc cho đến khi điều kiện dừng thỏa mãn.

Cách tiếp cận này sử dụng chiến lược chia để trị trong quá trình phân cụm.

3.3 Thuật toán

Thuật toán Birch nhận vào tập N dữ liệu đầu vào, dưới dạng vector, và số cụm mong muốn K, thuật toán sẽ xử lý qua 4 bước như sau :

- Bước 1 : Xây dựng cây đặc tính (CF) từ các điểm dữ liệu. Đây là cây cân bằng được định nghĩa như sau :
 - Cây CF là cây cân bằng nhằm để lưu trữ các đặc trưng của cụm. Cây CF chứa các nút trong và nút lá. Nút trong lưu trữ tổng các đặc trưng cụm của các nút con của nó. Một cây CF được đặc trưng bởi 2 tham số :
 - * Yếu tố nhánh (B) : Nhằm xác định số tối đa các nút con của mỗi nút trong của cây
 - * Yếu tố ngưỡng (T) : Khoảng cách tối đa giữa bất kỳ một cặp đối tượng trong nút lá của cây, khoảng cách này còn được gọi là đường kính của các cụm con, được lưu lại tại các nút lá.
- Bước 2 (không bắt buộc) : Thuật toán quét qua toàn bộ node lá từ cây CF ban đầu để xây dựng cây CF nhỏ hơn, xóa nhiễu và gom các cụm gần lại với nhau.
- Bước 3 : Một thuật toán gom cụm cho trước sẽ được sử dụng để gom tất cả các node lá. Ở đây, thuật toán phân cụm phân cấp thực thi trực tiếp vào các cụm con dựa trên vector CF của nó. Người dùng có thể chỉ định rõ số cụm mong muốn hoặc threshold cho các cụm. Sau bước này sẽ tạo thành các cụm biểu diễn gần đúng sự phân bố của dữ liệu ban đầu.
- Bước 4 (không bắt buộc) : Các tâm cụm tại bước 2 sẽ được điều chỉnh để tạo ra các cụm hợp lý hơn. Bước này còn có thể thêm hoạt động loại bỏ các điểm ngoại lai.

3.4 Ứng dụng

- BIRCH được sử dụng để phân loại với các tập dữ liệu rất lớn



- BIRCH tỏ ra hiệu quả hơn so với các thuật toán khác khi có thể xử lý dữ liệu lớn với không gian bộ nhớ hạn chế
- BIRCH hoạt động chính xác hơn nếu threshold đầu vào không quá lớn, thuật toán sẽ hoạt động nhanh hơn nếu threshold hợp lý với dữ liệu

Ưu điểm của BIRCH :

- Sử dụng cấu trúc cây CF làm cho thuật toán BIRCH có tốc độ thực hiện PCDL nhanh và có thể áp dụng đối với các tập dữ liệu lớn, BIRCH đặc biệt hiệu quả khi áp dụng với tập dữ liệu tăng trưởng theo thời gian
- BIRCH chỉ duyệt toàn bộ dữ liệu một lần với một lần quét thêm tùy chọn, nghĩa là độ phức tạp của nó là $O(n)$

Nhược điểm của BIRCH :

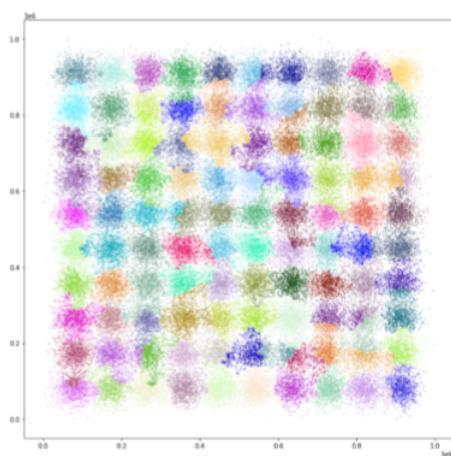
- Chất lượng của các cụm được khám phá không tốt
- Nếu BIRCH sử dụng khoảng cách Euclidean nó thực hiện tốt chỉ với các dữ liệu số. Mặt khác tham số của T có ảnh hưởng rất lớn đến kích thước và tính tự nhiên của cụm
- BIRCH không thích hợp với dữ liệu đa chiều

3.5 Thực nghiệm và đánh giá

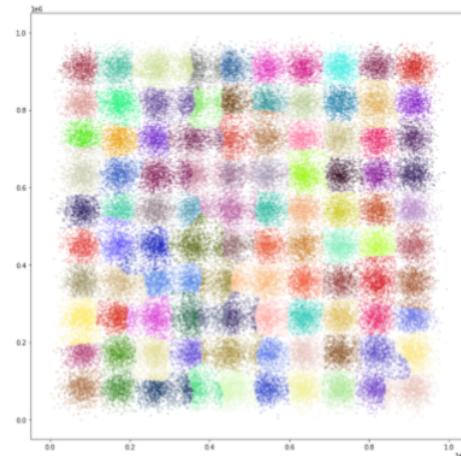
Thư viện ScikitLearn của Python có hỗ trợ công cụ BiRCH đã được tối ưu hóa. Ta sẽ thực nghiệm với công cụ này trên các tập dữ liệu Birch 1, Birch 2, Birch 3

3.5.1 Thủ nghiệm với tập dữ liệu Birch 1

Với tập dữ liệu này ta sẽ đặt số lượng các cụm tìm kiếm là 100

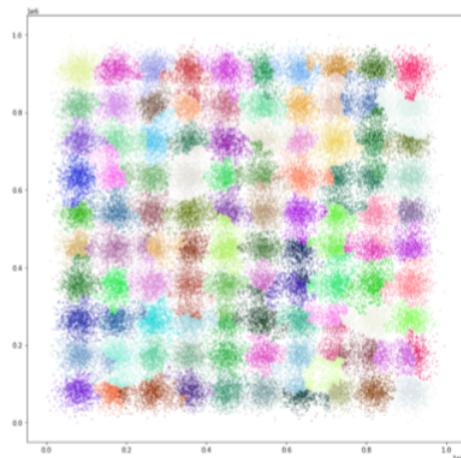


(a) branching factor = 100, threshold = 2000

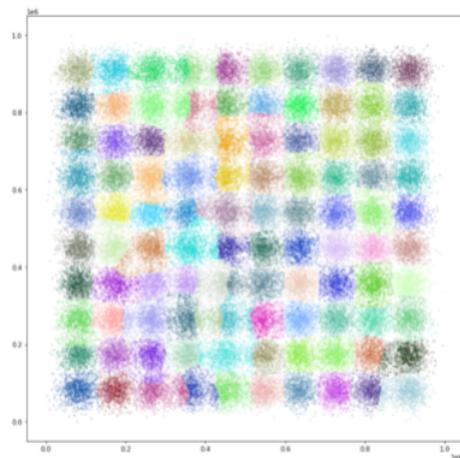


(b) branching factor = 100, threshold = 30000

Hình 8: Kết quả chạy BIRCH trên tập Birch1



(c) branching factor = 1000, threshold = 2000



(d) branching factor = 1000, threshold = 30000

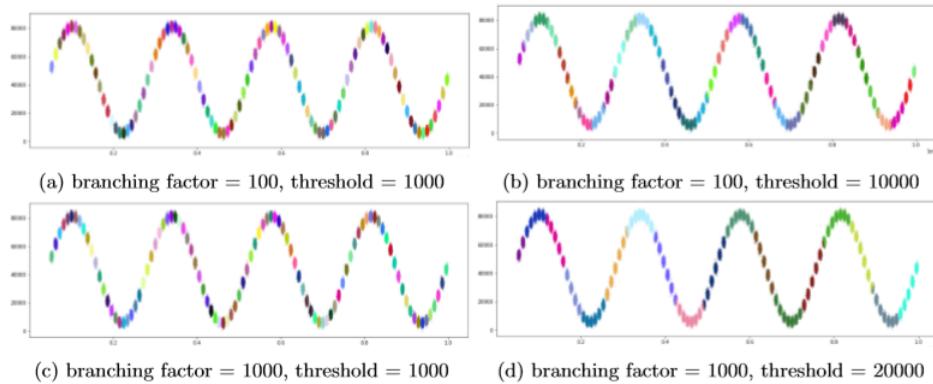
Hình 9: Kết quả chạy BIRCH trên tập Birch1

Số cụm	<i>branchingfactor</i>	<i>threshold</i>	Thời gian chạy
100	100	2000	67,18838460000006s
100	100	30000	1.66429499998674s
100	1000	2000	59.85691150000093s
100	1000	30000	1.335971399998348s

Hình 10: Thời gian xử lý tập Birch1 với các thông số tương ứng

3.5.2 Thử nghiệm với tập dữ liệu Birch 2

Với tập dữ liệu này, ta sẽ đặt số lượng các cụm tìm kiếm là 100



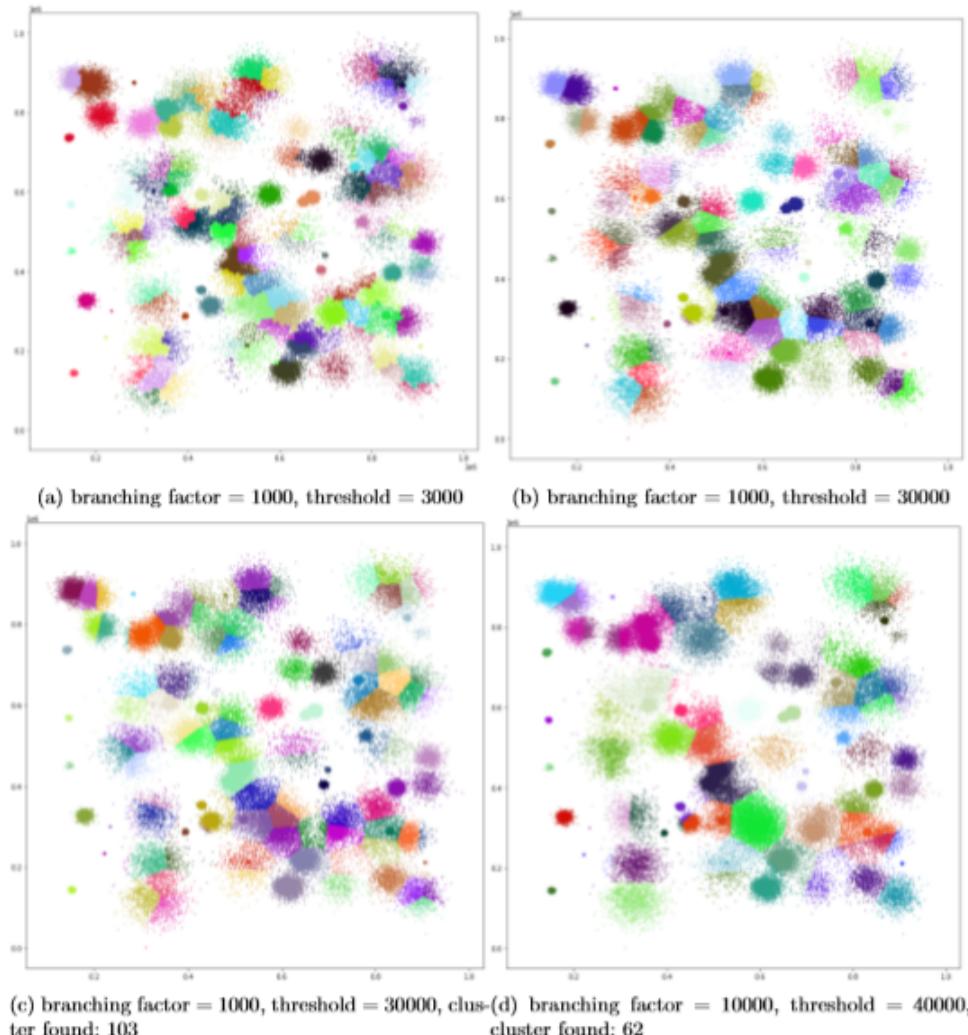
Hình 11: Kết quả chạy BIRCH trên tập Birch2

Số cụm	<i>branchingfactor</i>	<i>threshold</i>	Thời gian chạy
100	100	1000	3.696151700001792s
37	100	10000	1.218995600001985s
100	1000	1000	3.278275900001533s
18	1000	20000	1.110024900001008s

Hình 12: Thời gian xử lý tập Birch2 với các thông số tương ứng

3.5.3 Thử nghiệm với tập dữ liệu Birch 3

Với tập dữ liệu này, ta sẽ thử nghiệm đặt số lượng các cụm tìm kiếm là 100 và không đặt cụ thể.



Hình 13: Kết quả chạy BIRCH trên tập Birch3

Số cụm	<i>branchingfactor</i>	<i>threshold</i>	Thời gian chạy
100	1000	3000	12.003722600000003s
100	1000	30000	1.20071200000001s
103	1000	30000	1.1790667000000212s
62	10000	40000	1.1575862999999345s

Hình 14: Thời gian xử lý tập Birch3 với các thông số tương ứng



3.5.4 Nhận xét và đánh giá

Ta có thể thấy thuật toán chịu ảnh hưởng rất lớn từ việc lựa chọn chỉ số Threshold. Nếu threshold quá nhỏ, thuật toán trở nên overfit, không chính xác nữa, thời gian chạy chậm hơn. Nếu threshold quá cao, thuật toán kiểm được ít cụm hơn dự tính, thời gian chạy nhanh hơn. Ngược lại, chỉ số branching factor ảnh hưởng thuật toán hơn.

Đối với bộ dữ liệu Birch1, threshold thích hợp là 30000 và branching factor là 1000 vì khi đó, thuật toán chạy nhanh và các cụm được phân tốt hơn.

Đối với bộ dữ liệu Birch2, threshold 30000 lại không thích hợp, vì khi đó thuật toán tìm ra ít cụm hơn dự đoán.

Đối với bộ dữ liệu Birch3, threshold thích hợp là 30000 và branching factor là 1000 vì khi đó, thuật toán chạy nhanh và các cụm được phân tốt hơn.

4 CLARANS

4.1 Giới thiệu

4.1.1 Giải thuật PAM

PAM (Partitioning Around Medoids) là một giải thuật phân cụm dữ liệu k-medoid. Để tìm ra k cụm, PAM sẽ tìm ra một điểm biểu diễn cụm dữ liệu đó, gọi là medoid. Một khi đã chọn được điểm medoid, các điểm chưa được chọn sẽ nhóm với các điểm đã được chọn sao cho nó giống với điểm được chọn nhất. Ta nói một điểm O_j chưa được chọn thuộc về cluster thể hiện bởi điểm được chọn O_m khi $d(O_j, O_m) = \min_{O_e}(O_j, O_m)$ trong đó \min_{O_e} thể hiện khoảng cách ngắn nhất giữa tập các medoid và $d(O_1, O_2)$ thể hiện khoảng cách giữa 2 điểm O_1 và O_2 . Chất lượng của một cụm sẽ được tính toán là trung bình khoảng cách giữa các điểm của cụm với medoid của nó.

Giải thuật PAM sẽ bắt đầu bằng việc chọn k điểm phân biệt, Sau đó tại mỗi bước, một lần hoán vị giữa một điểm O_m được chọn và một điểm O_p không được chọn sẽ được thực hiện nếu như việc hiện thức đó gia tăng chất lượng phân cụm.

Giải thuật PAM cần phải tính toán tất cả chi phí của mọi cặp điểm được chọn và không được chọn để tìm ra chi phí thấp nhất. Vậy nên chi phí sẽ cần là $O(k(n-k)^2)$ trong lỗi lặp. Do đó, giải thuật này có thể hoạt động tốt với các tập dữ liệu nhỏ, nhưng lại khác chậm chạp với các tập dữ liệu vừa và lớn.



4.1.2 Giải thuật CLARA

CLARA (Clustering LARge Applications) là một giải thuật phân cụm k-medoid làm việc dựa trên việc lấy mẫu. Thay vì tìm các medoid cho toàn bộ tập dữ liệu, CLARA sẽ lấy ra một mẫu dữ liệu, áp dụng PAM trên mẫu lấy ra và tìm các medoid của mẫu. Nếu ta lấy mẫu một cách ngẫu nhiên thì các medoid của mẫu có thể xấp xỉ là medoid của toàn bộ tập dữ liệu. Ta có thể nâng cao chất lượng bằng cách lấy mẫu nhiều lần và xuất ra kết quả phân cụm có chất lượng cao nhất trên toàn tập dữ liệu. Thủ nghiệm cho thấy rằng 5 lần lấy mẫu có kích thước $40+2k$ sẽ cho ra kết quả là chấp nhận được.

Giải thuật CLARA được hiện thực như sau:

1. Lặp lại 5 lần, mỗi lần thực hiện như sau
2. Lấy ngẫu nhiên mẫu có kích thước $40+2k$ từ toàn bộ tập dữ liệu, hiện thực giải thuật PAM trên mẫu này.
3. Với mỗi điểm O_j trên tập dữ liệu, xác định medoid nào tương tự O_j nhất.
4. Xác định độ khác nhau trung bình giữa các cụm tìm được ở bước trước, nếu nó nhỏ hơn chỉ số hiện tại, đây sẽ là chỉ số tốt nhất hiện tại và các medoid tìm được sẽ là các medoid tốt nhất tìm được trong số các lần tìm được trước đó tới lần này.
5. Quay lại bước lặp 1.

Đối với CLARA, áp dụng PAM trên mỗi mẫu lấy có kích thước mẫu nên sẽ có độ phức tạp $O(k(40 + k)^2 + k(n - k))$. Do đó, giải thuật CLARA sẽ hiện quả hơn PAM đối với các giá trị n lớn (tập dữ liệu có kích thước lớn).

4.1.3 Giải thuật CLARANS

CLARANS (Clustering Larger Applications based on RANdomized Search) cũng là một giải thuật thuộc nhóm k-medoid. Giải thuật CLARANS sẽ cố gắng phân cụm n điểm dữ liệu thành k cụm một cách tốt nhất với k biết trước. Các giải thuật PAM, CLARA, CLARANS là các biến thể mà ta có thể giải thích và so sánh chúng thông qua các đồ thị.

Cho n điểm, công việc tìm k medoid có thể được coi như việc tìm trong một đồ thị. Trong đồ thị $G_{n,k}$, một node được thể hiện bởi tập k điểm $\{O_{m_1}, \dots, O_{m_k}\}$, là các điểm được chọn làm medoid. Hai node $S_1 = \{O_{m_1}, \dots, O_{m_k}\}$ và $S_2 = \{O_{l_1}, \dots, O_{l_k}\}$ được gọi là hàng xóm với nhau nếu chúng khác nhau đúng một phần tử, hay $|S_1 \cap S_2| = k - 1$.



Rõ ràng có thể thấy PAM chính là việc tìm kiếm một cực tiểu trên đồ thị $G_{n,k}$. Tại mỗi bước, tất cả các hàng xóm sẽ được xem xét. Node hiện tại sẽ được thay thế bởi hàng xóm có độ giảm chi phí lớn nhất. Và dễ thấy rằng mỗi node sẽ tồn k(n-k), việc xét tất cả các node này là rất tốn thời gian.

Dối với giải thuật CLARA, nó sẽ xem xét với số lượng ít hơn các hàng xóm và giới hạn việc tìm kiếm trên đồ thị con của $G_{n,k}$. Tuy nhiên, một vấn đề dễ thấy chính là nếu cực tiểu của đồ thị $G_{n,k}$ không nằm trong đồ thị con được chọn, nó sẽ không bao giờ được tìm thấy trên đồ thị được chọn. Do đó ta cần phải lấy mẫu nhiều lần để có tăng khả năng chính xác hơn.

Giải thuật CLARANS cũng tương tự với CLARA, không kiểm tra với mọi hàng xóm của một node. Tuy nhiên, CLARANS sẽ không giới hạn việc tìm kiếm trong một đồ thị con, nó sẽ tìm kiếm trong đồ thị gốc. Khác với PAM, CLARANS chỉ kiểm tra trên một mẫu hàng xóm của một node. Khác với CLARA sẽ lấy mẫu tại thời điểm ban đầu của việc tìm kiếm, CLARANS sẽ lấy một mẫu của các hàng xóm trong mỗi bước tìm kiếm. Điều này khiến cho việc tìm kiếm không bị giới hạn trong một vùng ngắn.

4.2 Hiện thực

1. Nhập vào 2 tham số *numlocal* và *maxneighbor*. Khởi tạo *i* bằng 1, *mincost* là một số rất lớn.
2. Đặt *current* là một node bất kỳ của $G_{n,k}$.
3. Đặt *j* bằng 1.
4. Xét một hàng xóm ngẫu nhiên *S* của *current*, và tính toán chi phí khác nhau giữa 2 node.
5. Nếu *S* có chi phí thấp hơn, đặt *current* thành *S* và quay lại bước 3.
6. Nếu không, tăng *j* lên 1. Nếu *j* $\leq maxneighbor$, đi tới bước 4.
7. Nếu *j* $> maxneighbor$, so sánh chi phí của *current* với *mincost*, nếu nó nhỏ hơn thì *mincost* sẽ là chi phí của *current* và *bestnode* sẽ là *current*.
8. Tăng *i* lên 1. Nếu *i* $> numlocal, xuất *bestnode* và kết thúc. Nếu không, quay lại bước 2.$

Bên cạnh số lượng cụm muốn phân, CLARANS còn có 2 tham số: số lượng tối đa các hàng xóm được xét (*maxneighbor*) và số lượng tối đa các cực tiểu địa

phương (*localminma*) có thể đạt được (*numlocal*). Giá trị *maxneighbor* càng lớn, CLARANS càng gầm PAM và thời gian tìm ra cực tiểu địa phương càng lâu. Nhưng chất lượng của các cực tiểu địa phương càng tốt và càng có ít số lượng cực tiểu địa phương cần đạt được càng ít.

Để xác định được các giá trị phù hợp cho 2 tham số kể trên, như các phương pháp ngẫu nhiên khác, ta phải phụ thuộc vào thử nghiệm.

4.3 Ứng dụng

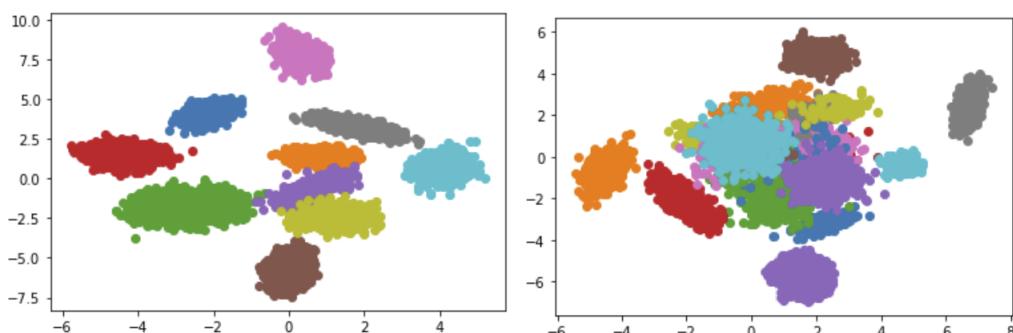
CLARANS được sử dụng để gom cụm các dữ liệu không gian, nó hạn chế được các điểm yếu của PAM và CLARA bằng cách cân bằng giữa chi phí tính toán và ảnh hưởng của việc lầu mẫu trên các tập dữ liệu.

4.4 Thực hiện và đánh giá

4.4.1 Thử nghiệm với tập dữ liệu sinh ra bởi HAWKS

HAWKS là một công cụ giúp sinh ra các tập dữ liệu nhân tạo, dùng để thực hiện các công việc phân cụm. Ta có thể chọn số cụm và một số thông số khác để sinh ra các tập thử nghiệm.

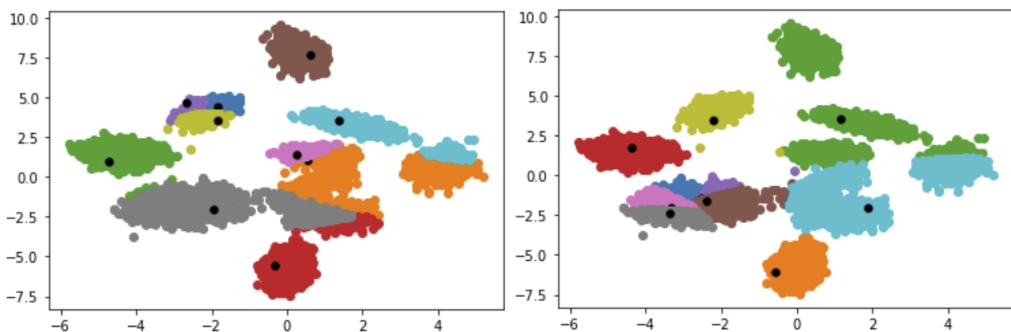
Ta sẽ lần lượt thử nghiệm CLARANS với 2 tập gồm 10000 điểm dữ liệu được phân thành 10 và 20 cụm bởi HAWKS. Phân bố điểm dữ liệu được cho bởi các hình dưới đây.



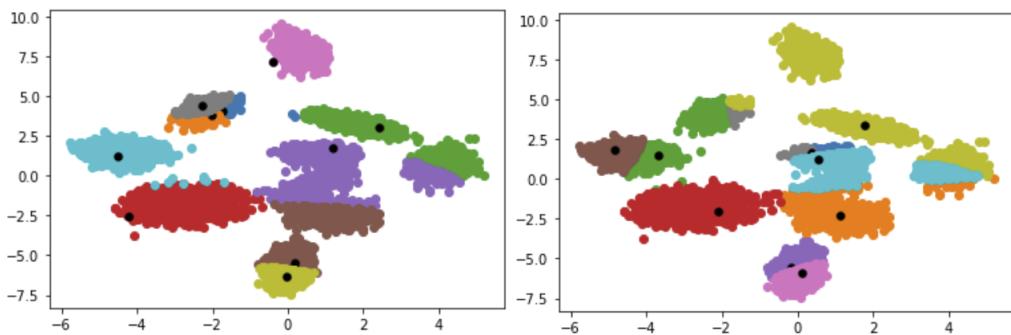
Hình 15: Phân bố của các tập dữ liệu sinh ra bởi HAWKS (Birch1 and Birch2)

Kết quả chạy CLARANS được cho bởi các hình ảnh và bảng dưới đây.

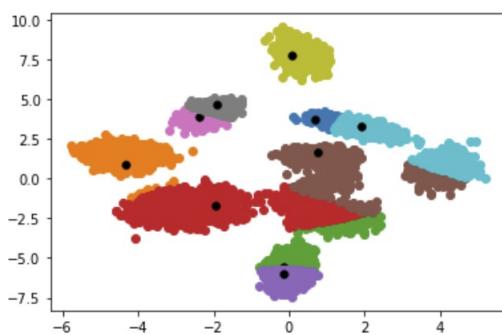
Kết quả chạy CLARANS trên tập 10 000 dữ liệu 10 cụm:



Hình 16: numlocal=3 maxneighbor=5 and numlocal=3 maxneighbor=10

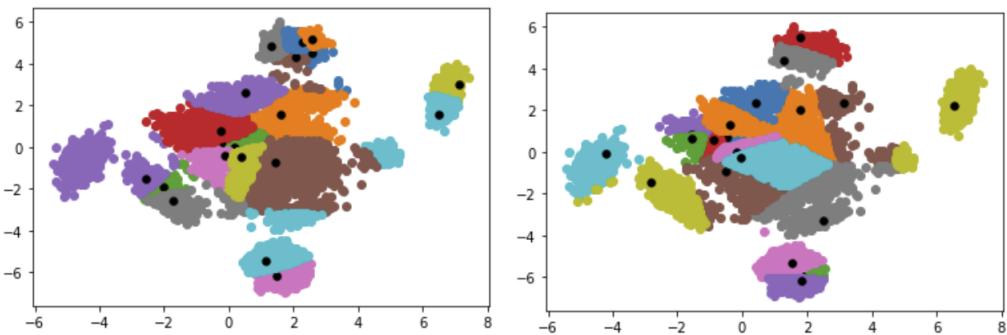


Hình 17: numlocal=4 maxneighbor=10 and numlocal=5 maxneighbor=5

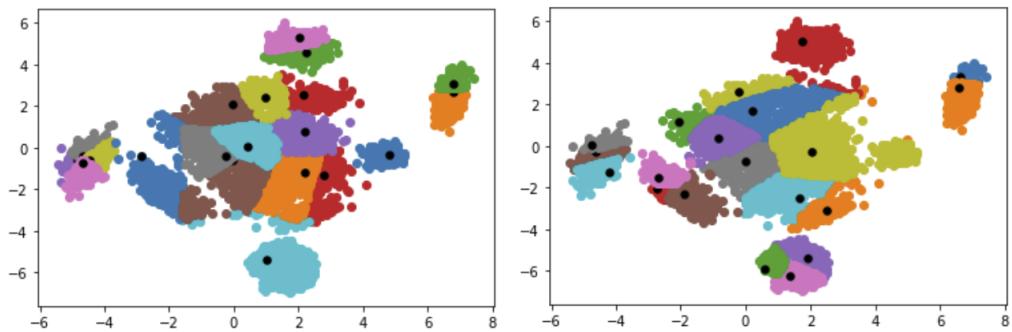


Hình 18: numlocal=5 maxneighbor=10

Kết quả chạy CLARANS trên tập 10 000 dữ liệu 20 cụm:



Hình 19: numlocal=3 maxneighbor=10 and numlocal=4 maxneighbor=10



Hình 20: numlocal=5 maxneighbor=5 and numlocal=5 maxneighbor=10

Số cụm	numlocal	maxneighbor	Thời gian chạy
10	3	5	3,6s
10	3	10	66,46s
10	4	10	82s
10	5	5	26,33s
10	5	10	231,7s
20	3	10	58,93s
20	4	10	47,81s
20	5	5	23,22s
20	5	10	156,79s

Hình 21: Thời gian xử lý các tập sinh bởi HAWKS với các thông số tương ứng



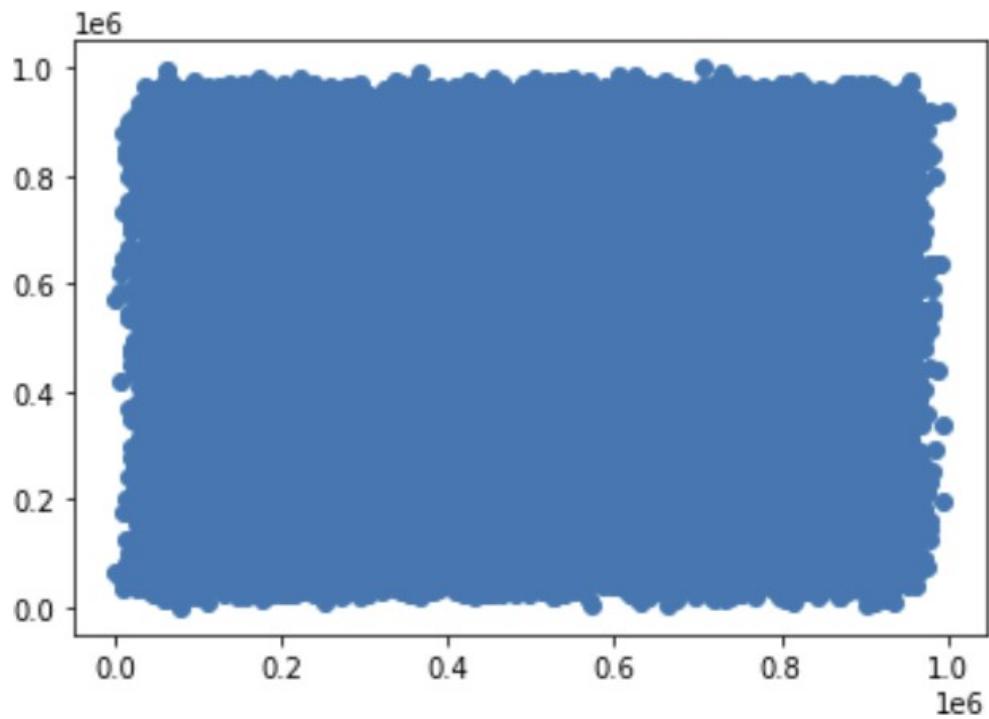
Nhận xét:

- Đối với chất lượng gom cụm, có thể thấy các cụm được phân ra chưa hợp lý về mặt nhìn nhận và trong so sánh với các cụm nguyên mẫu. Điều này xuất hiện trong cả 2 tập dữ liệu được sử dụng. Một trong những nguyên nhân bao gồm các tập sinh ra có nhiều cụm chồng lên nhau, và CLARANS không quá tuyệt vời để giải quyết vấn đề này. Khi giảm số lượng cụm được phân so với ground true table, kết quả chưa có dấu hiệu tốt lên.
- Về thời gian thực thi, có thể thấy thời gian khi ta thay đổi numlocal có tác động ít hơn so với khi ta thay đổi maxneighbor, maxneighbor tăng gấp đôi có thể khiến thời gian thực hiện tăng gần 10 lần trong khi tăng numlocal lên 1 chỉ khiến thời gian thực thi tăng lên khoảng 3 lần.

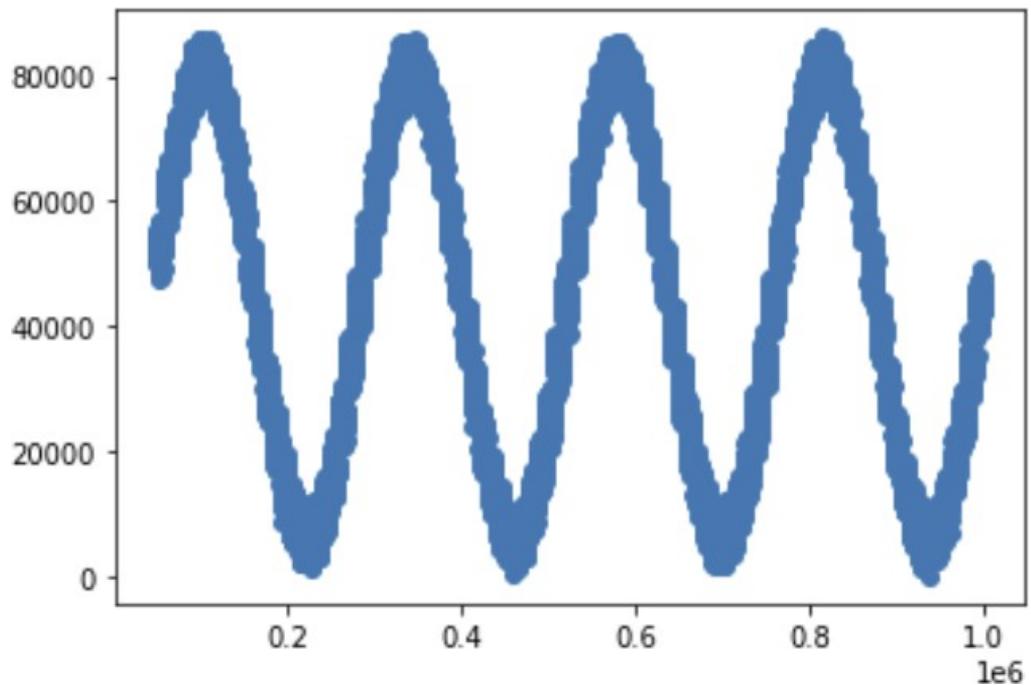
4.4.2 Thủ nghiệm với tập dữ liệu Birch

Ta sẽ thực hiện CLARANS với tập dữ liệu Birch1, Birch2, Birch3 được lấy ở trang cs.joensuu.fi. Đây là các dữ liệu dạng số thể hiện các điểm trên một trục toạ độ.

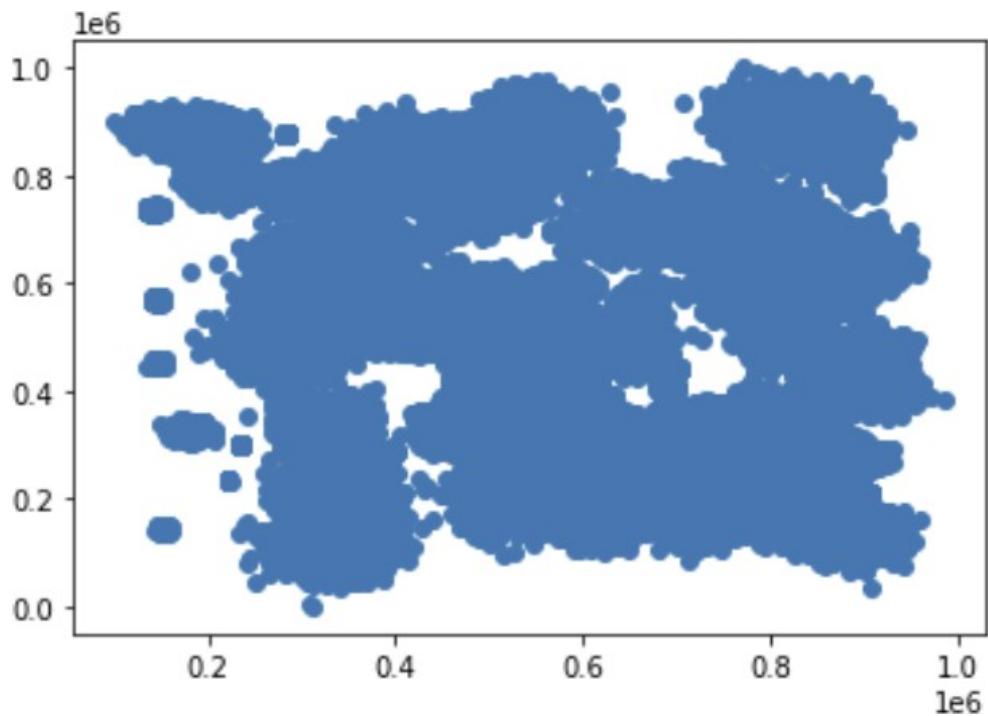
Các tập dữ liệu Birch1, Birch2, Birch3 gồm 100 000 điểm dữ liệu được phân thành 100 cụm với các điểm dữ liệu được phân bố như sau:



Hình 22: Phân bố của các tập dữ liệu Birch1



Hình 23: Phân bố của các tập dữ liệu Birch2

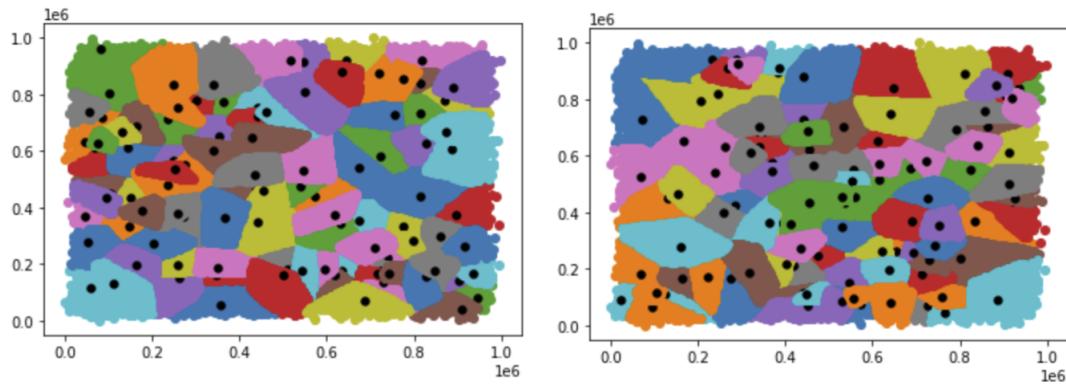


Hình 24: Phân bố của các tập dữ liệu Birch3

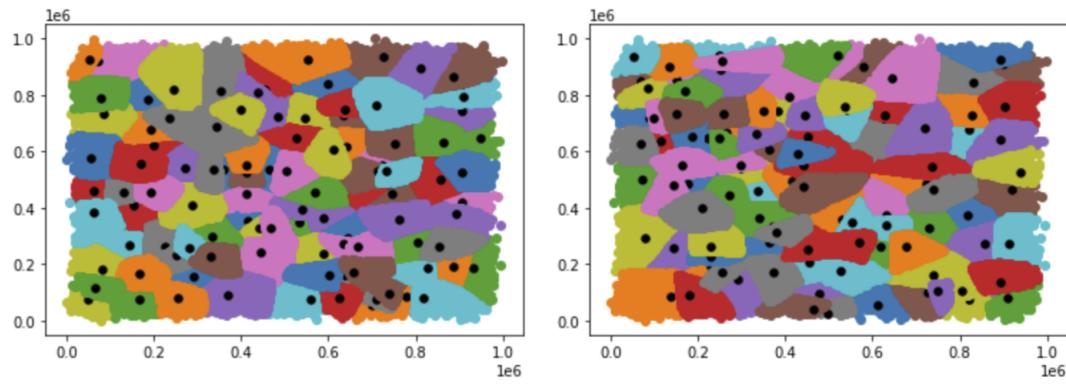
Do giải thuật này có thể có kết quả và thời gian thực thi khác nhau đối với các tham số đầu vào khác nhau, ta sẽ thực hiện thử nghiệm với các thông số đầu vào khác nhau.

4.4.2.a Thử nghiệm với tập dữ liệu Birch1

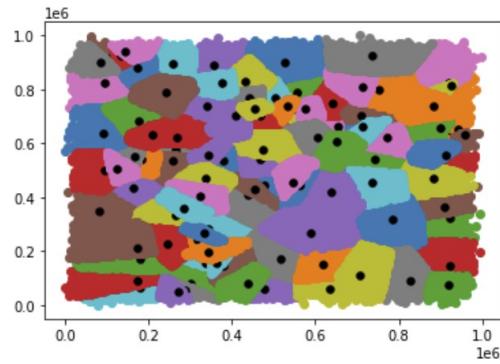
Các kết quả phân cụm và thời gian chạy của giải thuật đối với tập dữ liệu Birch1 được cho như các hình dưới đây.



Hình 25: numlocal=1 maxneighbor=5 and numlocal=1 maxneighbor=10



Hình 26: numlocal=2 maxneighbor=5 and numlocal=2 maxneighbor=10



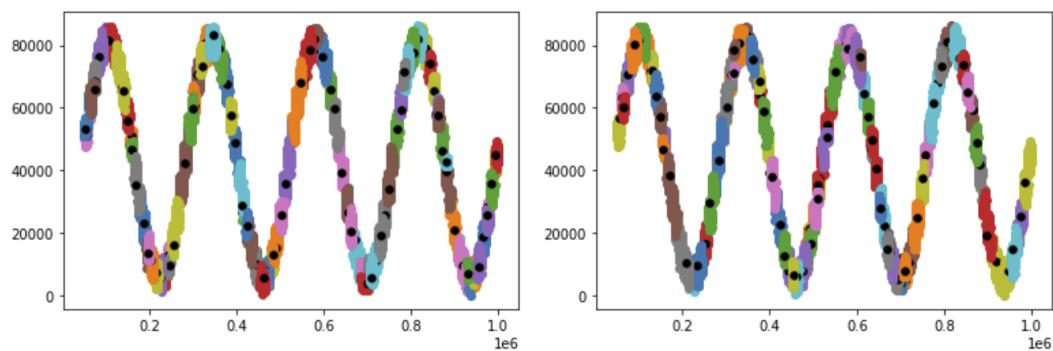
Hình 27: numlocal=5 maxneighbor=5

Số cụm	numlocal	maxneighbor	Thời gian chạy
100	1	5	60,74s
100	1	10	116,39s
100	2	5	220,44s
100	2	10	416,29s
100	5	5	287,39s

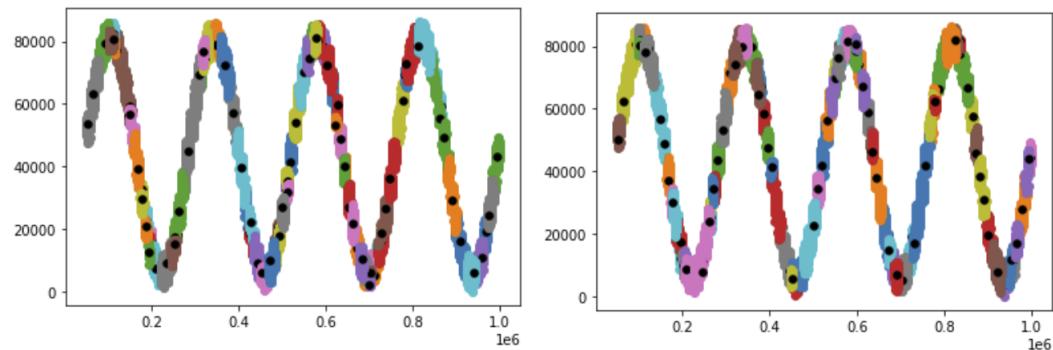
Hình 28: Thời gian xử lý tập Birch1 với các thông số tương ứng

4.4.2.b Thử nghiệm với tập dữ liệu Birch2

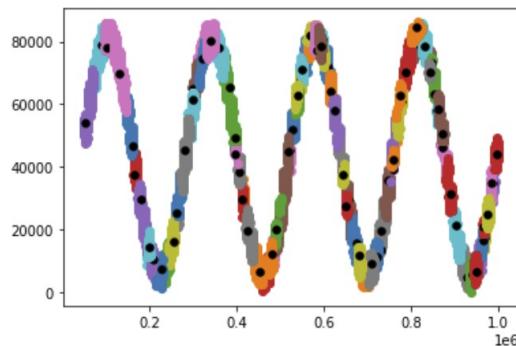
Các kết quả phân cụm và thời gian chạy của giải thuật đối với tập dữ liệu Birch2 được cho như các hình dưới đây.



Hình 29: numlocal=1 maxneighbor=5 and numlocal=1 maxneighbor=10



Hình 30: numlocal=2 maxneighbor=5 and numlocal=2 maxneighbor=10



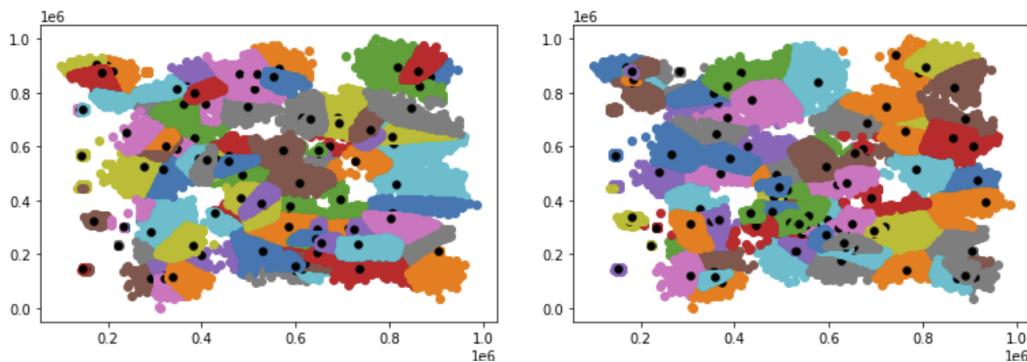
Hình 31: numlocal=5 maxneighbor=5

Số cụm	numlocal	maxneighbor	Thời gian chạy
100	1	5	64,22s
100	1	10	101,63s
100	2	5	118,49s
100	2	10	184,71s
100	5	5	327,37s

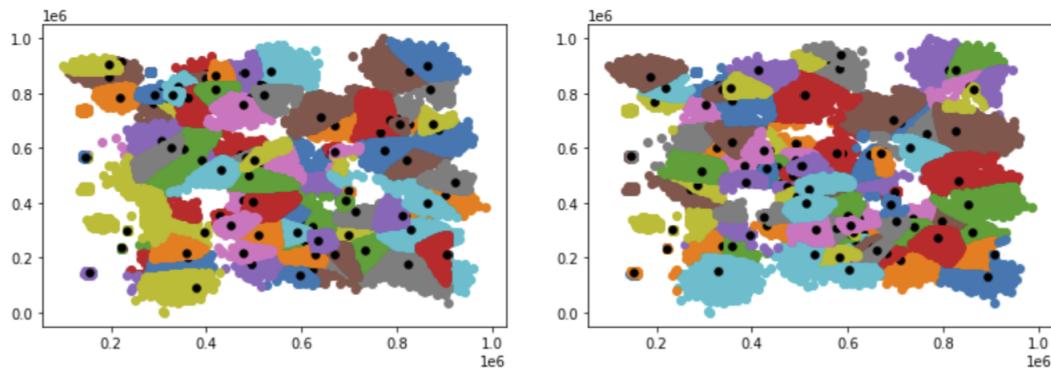
Hình 32: Thời gian xử lý tập Birch2 với các thông số tương ứng

4.4.2.c Thử nghiệm với tập dữ liệu Birch3

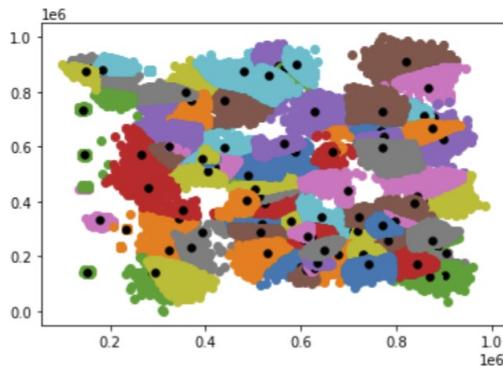
Các kết quả phân cụm và thời gian chạy của giải thuật đối với tập dữ liệu Birch3 được cho như các hình dưới đây.



Hình 33: numlocal=1 maxneighbor=5 and numlocal=1 maxneighbor=10



Hình 34: numlocal=2 maxneighbor=5 and numlocal=2 maxneighbor=10



Hình 35: numlocal=5 maxneighbor=5

Số cụm	numlocal	maxneighbor	Thời gian chạy
100	1	5	61,52s
100	1	10	171,78s
100	2	5	111,69s
100	2	10	390,3s
100	5	5	378,29s

Hình 36: Thời gian xử lý tập Birch3 với các thông số tương ứng

4.4.2.d Nhận xét

Đối với tập dữ liệu Birch1, có thể thấy, so với ground true table được cung cấp cho tập Birch1, kết quả tạo ra số cụm mong muốn tuy nhiên các medoid tìm được



bị lệch tương đối nhiều, các cụm có kích thước không đều.

Đối với các kết quả đạt được với tập dữ liệu Birch2, các cụm phân ra đạt được kết quả tốt hơn so với Birch1, các medoid tìm được đều nằm trên đường phân bố, kích thước các cụm tương đối đều nhau với các thông số nhập vào, tuy nhiên còn một số cụm bị dính với nhau.

Đối với tập dữ liệu Birch3, đối với các trường hợp với số lượng numlocal và maxneighbor nhỏ, các điểm outlier có xu hướng bị gộp vào với các cụm lớn hơn. Khi thiết lập tăng numlocal hoặc maxneighbor, các cụm được phân hoá chính xác hơn.

Điều đó cho thấy giải thuật CLARANS có thể tương đối nhạy cảm với các outliers và do đó cần thiết phải thiết lập số lượng numlocal cũng như maxneighbor tương đối để có kết quả chấp nhận được. Với số lượng numlocal hoặc maxneighbor thấp có thể dẫn đến các cụm phân ra không như mong muốn.

Về mặt thời gian thực thi, qua 3 kết quả của 3 tập dữ liệu, ta có thể thấy nó sẽ tăng gần như tương ứng tỉ lệ thuận khi ta tăng số lượng maxneighbor hay minlocal. Do đó, để đạt được tối ưu về mặt hiệu suất cũng như chất lượng kết quả, ta cần phải thực hiện nhiều lần trên từng tập dữ liệu cụ thể.

5 DBSCAN

5.1 Giới thiệu

(Density-based spatial clustering of applications with noise) tạm dịch là phân cụm không gian dựa trên mật độ cho các ứng dụng với nhiễu

Đây là thuật toán phân cụm dựa theo mật độ phi tham số: Với một tập các điểm đã cho trong không gian, thuật toán sẽ gom nhóm các điểm lân cận lại với nhau thành một nhóm, và đánh dấu là các điểm outlier nếu chúng nằm ở vùng có mật độ thấp, tách biệt với các điểm đã gom nhóm. DBSCAN là một trong những thuật toán phân cụm phổ biến nhất và cũng được trích dẫn nhiều nhất trong các tài liệu khoa học.

-Lý thuyết:Xét một tập hợp các điểm trong một số không gian thuộc hiện phân cụm. Ta có:

- ϵ là một tham số xác định bán kính của vùng lân cận với một điểm. Gọi là



ϵ -neighborhood.

- + MinPts: số lượng điểm ít nhất được yêu cầu nằm trong phạm vi ϵ của một điểm.
- Nếu điểm thỏa mãn điều kiện MinPts với thông số ϵ thì điểm này được gọi là core object
- Hình dạng vùng lân cận của một điểm được xác định dựa vào việc chọn hàm khoảng cách giữa hai điểm p và q, ký hiệu là distance (p,q). Ví dụ, nếu dùng khoảng cách Manhattan trong không gian 2D thì hình dạng vùng lân cận là hình chữ nhật.

-Các khái niệm quan trọng:

- Directly density-reachable (khả năng đạt được trực tiếp): q có thể đạt được trực tiếp từ p nếu q nằm trong phạm vi ϵ của p và p là core object.
- Density-reachable (khả năng đạt được): q có thể đạt được từ p khi và chỉ khi có một chuỗi các điểm p_1, \dots, p_n với $p_1 = p$ và $p_n = q$ sao cho p_{i+1} là directly density-reachable từ p_i theo thông số ϵ và MinPts.
- Density-connected (kết nối dựa trên mật độ): q được coi kết nối dựa trên mật độ với p sao cho cả q và p đều là density-reachable từ o theo thông số ϵ và MinPts.

5.2 Hiện thực

1. Chọn độ đo khoảng cách giữa hai điểm trong không gian là euclid.
2. Sử dụng KD-Tree để lưu trữ cũng như tìm kiếm các điểm (neighbours) nằm trong phạm vi có bán kính là ϵ của mỗi điểm trong không gian đã cho. Sử dụng cấu trúc lưu trữ thế này rất là hữu ích vì nó có thể tránh được không gian tìm kiếm lớn cũng như có thể áp dụng trên nhiều chiều.

KD-Node có hai giá trị trong đó key lưu giữ thứ tự insert vào nhầm phân biệt các node với nhau còn val lưu giữ tạo độ của node đó.

Trong cây KD-Tree có 3 hàm trong đó hàm insert để thêm một node vào cây, hàm range_search_result trả về kết quả khi gọi hàm range_search. Hàm range_search tìm kiếm các điểm thỏa mãn ϵ với độ đo là euclid.

3. Phân loại các điểm trong không gian thành core point, border point hay outlier point bằng cách so sánh tổng neighbours point của mỗi điểm với MinPts.

Ở class DBSCAN bao gồm:

- no_dimensions là số chiều của dữ liệu
- eps chính là ϵ đã đề cập ở lý thuyết
- min_points cũng là MinPts đã đề cập ở lý thuyết



- Ngoài ra em còn thêm giá trị là min_elements được hiểu như là số phần tử hay là điểm nhỏ nhất trong một cụm thì ta mới công nhận đó là một cụm hợp lệ. Việc định nghĩa thêm giá trị min_elements này giúp cho ta loại bỏ thêm những điểm outlier. Ta định nghĩa hàm neighbour_points trả về số lượng phần tử là "neighbours" của mỗi điểm từ tập dữ liệu. Việc này được sử dụng cho quá trình xác định core point, border point hoặc là outlier point.

4. Sử dụng Breadth First Search để gom cụm lại các điểm đã được phân loại từ bước 3.

5. Loại bỏ các điểm outlier bằng cách loại bỏ các điểm có label đã được đánh dấu là -1.

5.3 Ứng dụng

1. Thuật toán DBSCAN được sử dụng để tìm các liên kết và cấu trúc trong dữ liệu được sử dụng cho việc tìm kiếm các mẫu và dự đoán xu hướng.

2. Thuật toán DBSCAN có thể được sử dụng để tìm và phân loại các nguyên tử trong dữ liệu và được sử dụng trong X-ray crystallography.

3. Phát hiện bất thường bằng cách sử dụng kỹ thuật phân cụm DBSCAN để giám sát video giao thông giúp phát hiện vi phạm quy tắc, tai nạn, ...

5.4 Thực nghiệm và đánh giá

Lưu ý các kết quả đo thời gian dưới đây đều được tính dưới đơn vị là giây

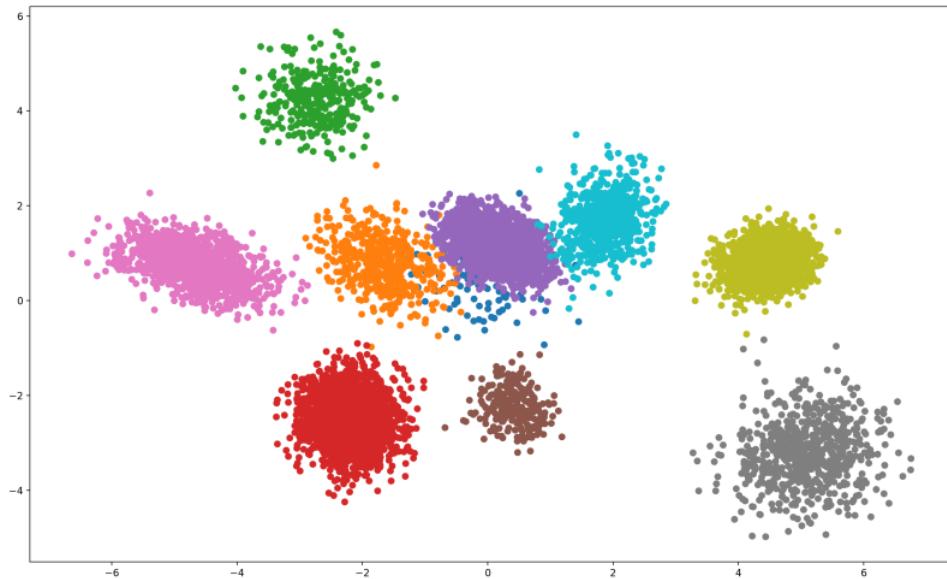
5.4.1 Thủ nghiệm với tập dữ liệu sinh ra bởi HAWKS

Tập dữ liệu chứa 10000 điểm và được chia thành 10 cụm

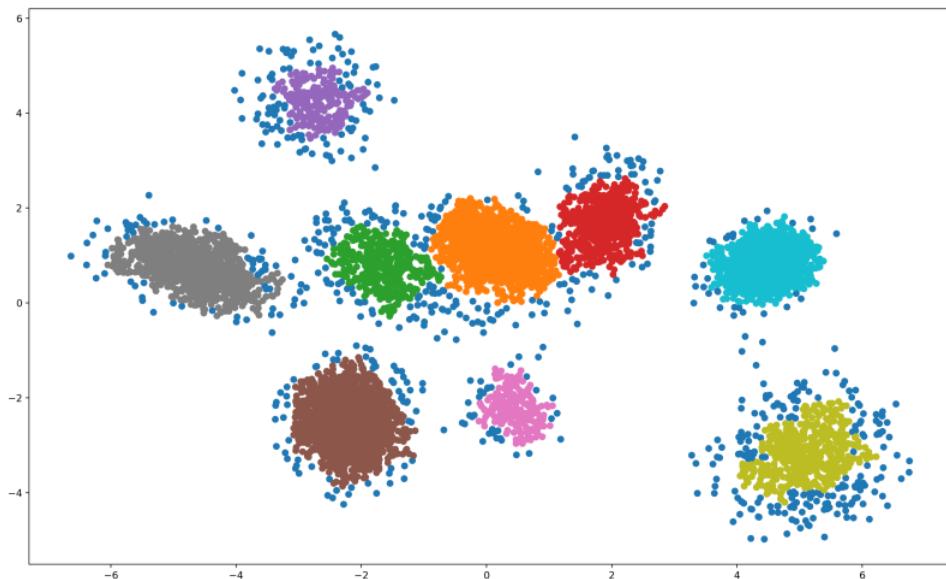
- Kết quả khi chạy với tham số đầu vào eps = 0.11, min_points = 3 và min_elements = 20

```
Thời gian để insert vào KDTree: 0.12623551900000018
Thời gian để tìm neighbours của mỗi điểm: 1.922881504
Thời gian để gom cụm các điểm: 0.07984817399999988
ARI: 0.9299380879164827
Thời gian để loại bỏ các điểm outlier: 0.00800072199999794
Tổng thời gian thực thi: 2.136965918999997
Tổng số core point và border point là: 9284
Tổng số outlier point là: 717
Tổng số cluster là: 9
```

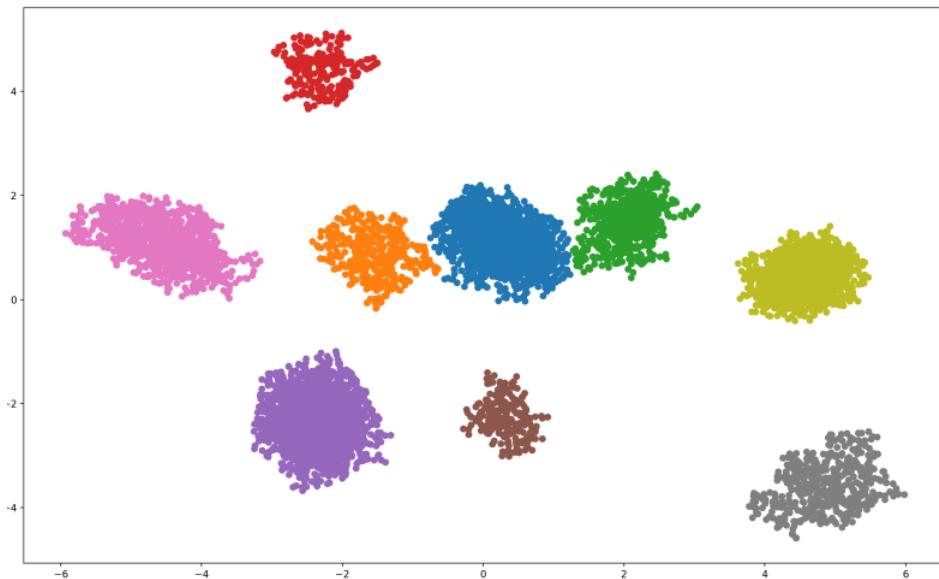
Hình 37: Kết quả khi chạy với DBSCAN



Hình 38: Kết quả đúng của tập dữ liệu HAWKS



Hình 39: Biểu đồ các cụm và những điểm outlier



Hình 40: Biểu đồ các cụm sau khi loại bỏ các điểm outlier

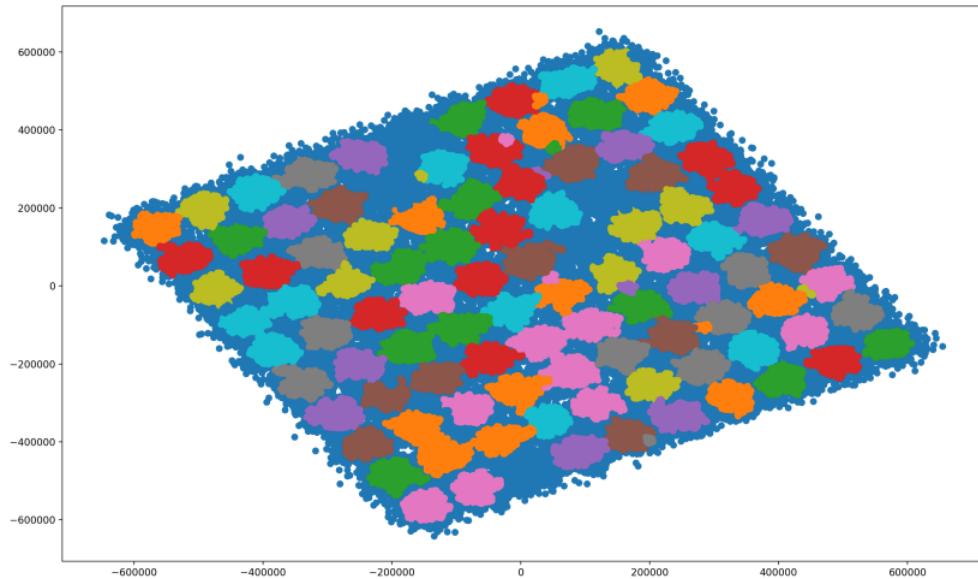
- Nhận xét: Nhìn chung đối tập dữ liệu mà các cụm ít có sự chồng lênh thế này thì DBSCAN cho ra kết quả khá tốt với ARI gần bằng 0.93, nhưng vẫn tồn tại một cụm mà DBSCAN không thể phát hiện được, do cụm xanh dương nằm sát với 2 cụm khác là cam và tím như ở (hình 21).

5.4.2 Thủ nghiệm với tập dữ liệu Birch1

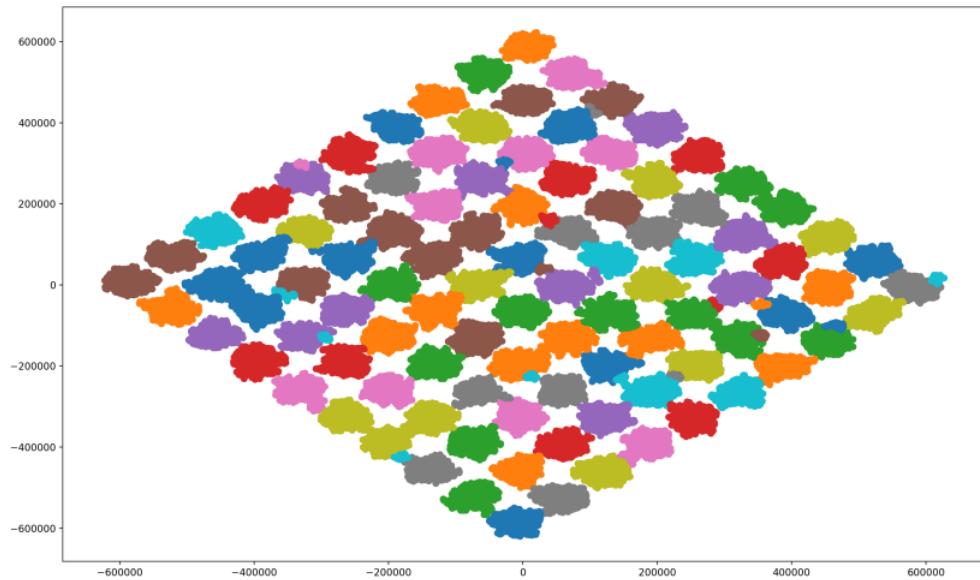
- Kết quả khi chạy với tham số đầu vào $\text{eps} = 3500$, $\text{min_points} = 4$ và $\text{min_elements} = 30$

```
Thời gian để insert vào KDTree: 2.634514587
Thời gian để tìm neighbours của mỗi điểm: 101.472943496
Thời gian để gom cụm các điểm: 0.21165749000000744
Thời gian để loại bỏ các điểm outlier: 0.029377032999988728
Tổng thời gian thực thi: 104.348492606
Tổng số core point và border point là: 76450
Tổng số outlier point là: 23550
Tổng số cluster là: 112
```

Hình 41: Kết quả trên tập dữ liệu Birch 1



Hình 42: Biểu đồ bao gồm cả các cụm và những điểm outlier



Hình 43: Biểu đồ các cụm sau khi loại bỏ các điểm outlier

- Nhận xét: Đối với tập dữ liệu Birch 1 này DBSCAN chưa cho thấy được những ưu điểm của mình. Mặc dù Khi nhìn vào (hình 27) ta có thể thấy rõ

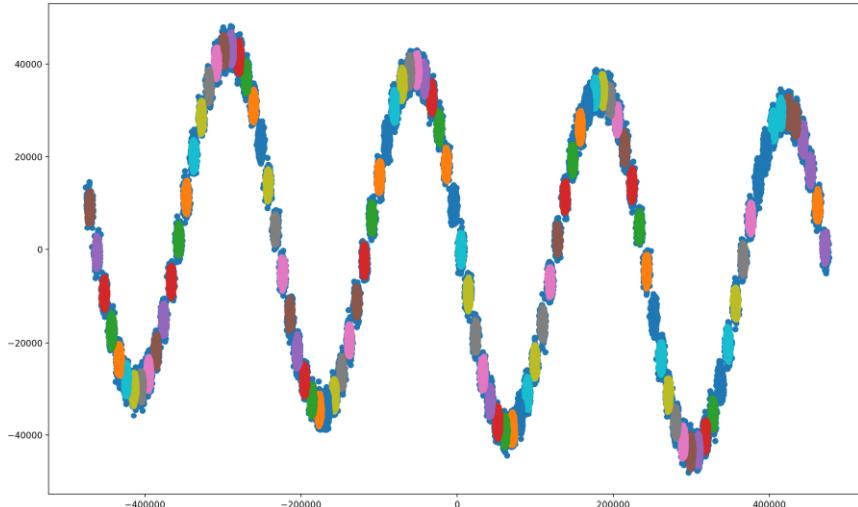
hơn được sự phân biệt giữa mỗi cụm trong không gian. Tuy nhiên vẫn chưa quá tốt khi vẫn còn một số cụm lớn vẫn còn dính liền nhau và có lác đác một số cụm nhỏ lẻ. Số điểm outlier được xác định cũng quá lớn làm dữ liệu bị mất đi khá nhiều. Nếu tăng min_points hay giảm eps thì sẽ xuất hiện nhiều điểm outlier dẫn đến làm mất đi quá nhiều dữ liệu còn ngược lại thì số cụm xác định lại giảm đi và mất tính chính xác.

5.5 Thủ nghiệm với tập dữ liệu Birch2

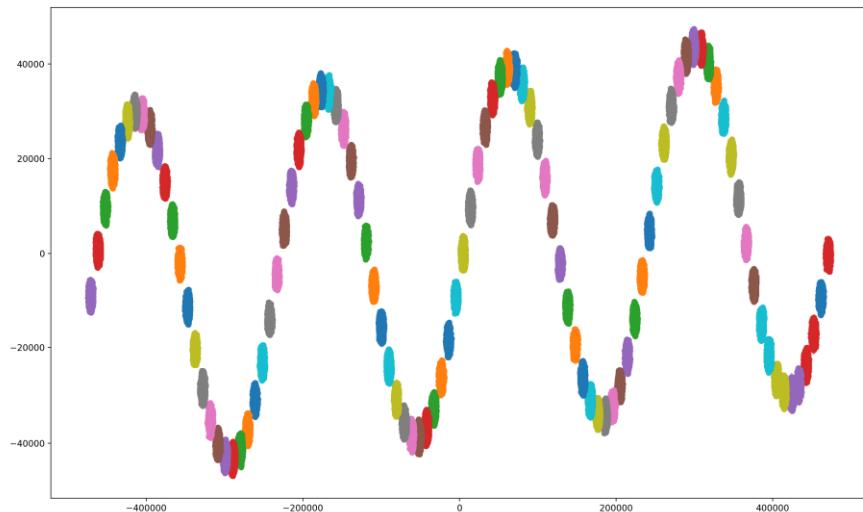
Kết quả khi chạy với tham số đầu vào $\text{eps} = 1000$, $\text{min_points} = 45$ và $\text{min_elemets} = 3$

```
Thời gian để insert vào KDTree: 3.497457486
Thời gian để tìm neighbours của mỗi điểm: 340.482054021
Thời gian để gom cụm các điểm: 1.918719904999989
Thời gian để loại bỏ các điểm outlier: 0.03439051800000925
Tổng thời gian thực thi: 345.93262193
Tổng số core point và border point là: 94570
Tổng số outlier point là: 5430
Tổng số cluster là: 100
```

Hình 44: Kết quả trên tập dữ liệu Birch 2



Hình 45: Biểu đồ bao gồm cả các cụm và những điểm outlier



Hình 46: Biểu đồ các cụm sau khi loại bỏ các điểm outlier

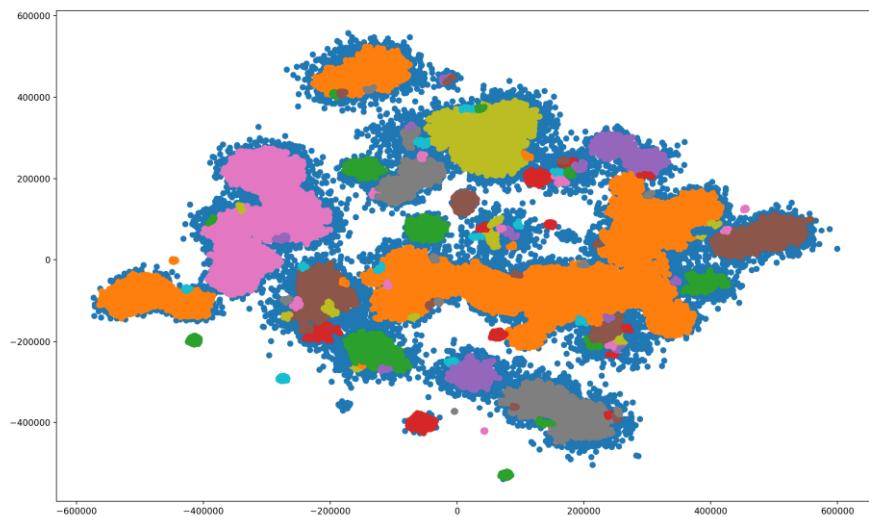
Nhận xét: Đối với tập dữ liệu Birch 2 này DBSCAN đã cho thấy ưu điểm của mình đối với mọi hình dáng của tập dữ liệu. DBSCAN đã giúp ta xác định được chính xác có 100 cụm và việc xác định outlier cũng hiệu quả, nhìn vào (hình 30) ta có thể thấy rõ được sự phân biệt giữa mỗi cụm. Tuy nhiên việc tìm kiếm neighbours mất khá nhiều thời gian.

5.6 Thủ nghiệm với tập dữ liệu Brich3

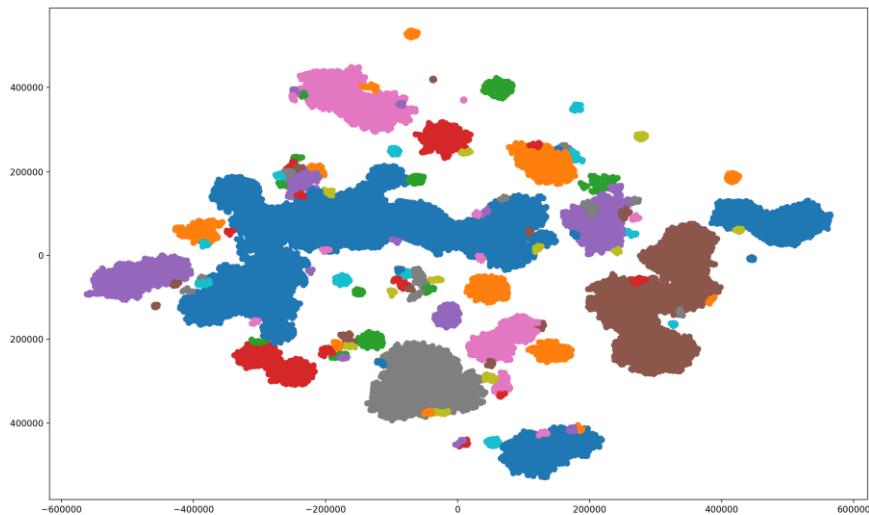
Kết quả khi chạy với tham số đầu vào $\text{eps} = 4000$, $\text{min_points} = 4$ và $\text{min_elements} = 10$

```
Thời gian để insert vào KDTree: 4.258862033
Thời gian để tìm neighbours của mỗi điểm: 165.851509857
Thời gian để gom cụm các điểm: 4.947063383999989
Thời gian để loại bỏ các điểm outlier: 0.034296153000013874
Tổng thời gian thực thi: 175.091731427
Tổng số core point và border point là: 94186
Tổng số outlier point là: 5814
Tổng số cluster là: 116
```

Hình 47: Biểu đồ Brich3



Hình 48: Biểu đồ các cụm sau khi loại bỏ các điểm outlier



Hình 49: Biểu đồ các cụm sau khi loại bỏ các điểm outlier

Nhận xét:

Đối với tập dữ liệu Birch 3 này DBSCAN không phát huy được tác dụng trong



việc phân cụm. Khi mà kết quả phân cụm ở (hình 33) chưa cho thấy được sự phân hóa rõ rệt giữa các cụm. Tuy nhiên ta có thể sử dụng giải thuật DBSCAN cho tập dữ liệu này để xác định nhiễu, khi đã loại bỏ các điểm outlier như trong (hình 32)

6 Tài liệu tham khảo