

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NHA TRANG
KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI BÁO CÁO
CẢI THIẾN HIỆU SUẤT CỦA THUẬT TOÁN GOM NHÓM
DỮ LIỆU KHÔNG GIAN NS-DBSCAN**

Giảng viên hướng dẫn: ThS. Nguyễn Thủy Đoan Trang

Sinh viên thực hiện:

- 1. Lê Văn Hòa - 65131052**
- 2. Nguyễn Gia Khánh - 65131461**
- 3. Nguyễn Thành Đạt - 65139021**

Lớp học phần: 65.CNTT-3

Khánh Hòa, tháng 01 năm 2026

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NHA TRANG
KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI BÁO CÁO
CẢI THIỆN HIỆU SUẤT CỦA THUẬT TOÁN GOM NHÓM
DỮ LIỆU KHÔNG GIAN NS-DBSCAN**

Giảng viên hướng dẫn: ThS. Nguyễn Thủy Đoan Trang

Sinh viên thực hiện:

- 1. Lê Văn Hòa - 65131052**
- 2. Nguyễn Gia Khánh - 65131461**
- 3. Nguyễn Thành Đạt - 65139021**

Lớp học phần: 65.CNTT-3

Khánh Hòa, tháng 01 năm 2026

Mục lục

I. GIỚI THIỆU	6
1. Đặt vấn đề	6
2. Thách thức nghiên cứu	6
3. Mục tiêu và Giải pháp đề xuất	7
II. CÁC KIẾN THỨC CƠ SỞ	8
1. Mô hình hóa không gian mạng	8
2. Khoảng cách trong Không gian mạng	8
3. Các định nghĩa Phân cụm	9
III. ĐỀ XUẤT GIẢI PHÁP TỐI ƯU HÓA THUẬT TOÁN PHÂN CỤM TRONG KHÔNG GIAN MẠNG	10
1. Cơ chế hoạt động của thuật toán NS-DBSCAN nguyên bản	10
1.1. Giai đoạn 1: Xây dựng trật tự mật độ	10
1.2. Giai đoạn 2: Hình thành Cụm	11
2. Các chiến lược tối ưu hóa trọng thuật toán cải tiến iNS-DBSCAN	11
2.1. Chiến lược Lược bỏ Cạnh dựa trên Ràng buộc Độ dài	11
2.1.1. Phân tích điểm nghẽn của thuật toán gốc	11
2.1.2. Cơ chế của thuật toán cải tiến	12
2.2. Tối ưu hóa cấu trúc bảng trật tự mật độ	13
2.2.1. Phân tích điểm nghẽn trong thuật toán gốc	13
2.2.2. Cơ chế của thuật toán cải tiến	14
2.3. Tích hợp quy trình xác định nhiễu ngầm định	15
2.3.1. Phân tích điểm nghẽn trong thuật toán gốc	15
2.3.2. Cơ chế của thuật toán cải tiến	15
3. Mã giả và quy trình thực thi thuật toán cải tiến	16
IV. Đánh giá và So sánh Hiệu năng Thực thi	24
1. Thiết lập thực nghiệm và Phương pháp đánh giá	24
2. Phân tích và Nhận xét kết quả	25
V. Kết luận và Hướng phát triển	27
1. Kết luận	27
2. Hướng phát triển	27

VI. LỜI CẢM ƠN	29
Tài liệu tham khảo	30

BẢNG PHÂN CÔNG NHIỆM VỤ THÀNH VIÊN

Họ và tên	Mức độ	Nội dung phân công
Lê Văn Hòa	100%	Nghiên cứu tổng quan đề tài, tìm hiểu thuật toán NS-DBSCAN và iNS-DBSCAN. Viết báo cáo, xây dựng và chạy thuật toán, kiểm tra câu hỏi.
Nguyễn Gia Khánh	100%	Viết tool chạy dữ liệu, kiểm tra định dạng báo cáo, test dữ liệu, soạn câu hỏi.
Nguyễn Thành Đạt	100%	Soạn slide, chỉnh sửa chương 4.

Danh sách hình vẽ

III.1 Minh họa điều kiện	12
III.2 Mô tả thuật toán LSPD cải tiến	17
III.3 Minh họa mở rộng với Central point: P5	18
III.4 Thuật toán sắp xếp mật độ cải tiến (Improved Generating Density Ordering)	19
III.5 Bảng sắp xếp mật độ	20
III.6 Thuật toán hình thành cụm cải tiến (Improved Cluster Formation)	22
III.7 Minh họa	23
IV.1 So sánh thuật toán gốc và cải tiến – Dữ liệu Cần Thơ	25

Danh sách bảng

III.1 Pseudocode of local shortest-path distance (LSPD) Algorithm.	12
III.2 Pseudocode of Generating Density Ordering Table and Graph.	14
III.3 Pseudocode of Forming Clusters.	15

Chương I

GIỚI THIỆU

1 Đặt vấn đề

Sự phát triển bùng nổ của công nghệ định vị toàn cầu (GPS) và các thiết bị di động thông minh đã tạo ra nguồn dữ liệu khổng lồ chứa thông tin không gian. Thực tế này đặt ra nhu cầu cấp thiết về các phương pháp khai phá dữ liệu hiệu quả, trong đó kỹ thuật phân cụm dữ liệu (Data Clustering) đóng vai trò then chốt nhằm phát hiện các mẫu tiềm ẩn và tri thức hữu ích từ dữ liệu thô.

Tuy nhiên, đa số các thuật toán phân cụm truyền thống hiện nay hoạt động dựa trên giả định không gian Euclid, nơi khoảng cách giữa hai điểm được tính bằng đường thẳng. Cách tiếp cận này bỏ qua các ràng buộc vật lý và cấu trúc hạ tầng thực tế như sông ngòi, tòa nhà hay mạng lưới đường sá. Sự đơn giản hóa này dẫn đến sai lệch nghiêm trọng trong các ứng dụng đô thị; ví dụ điển hình là việc gộp hai vị trí nằm đối diện nhau qua bờ sông vào cùng một cụm dù không có đường đi trực tiếp kết nối chúng. Do đó, việc chuyển dịch mô hình nghiên cứu sang Không gian Mạng (Network Space), nơi khoảng cách được đo bằng độ dài đường đi ngắn nhất trên đồ thị thực tế, là bước tiến bắt buộc để đảm bảo độ chính xác về mặt ngữ nghĩa của kết quả phân tích.

2 Thách thức nghiên cứu

Trong các phương pháp phân cụm dựa trên mật độ, thuật toán DBSCAN được đánh giá cao nhờ khả năng phát hiện các cụm có hình dạng bất kỳ và loại bỏ nhiễu hiệu quả. Kế thừa ưu điểm này, Wang và cộng sự đã phát triển thuật toán NS-DBSCAN, sử dụng kỹ thuật tính khoảng cách đường đi ngắn nhất cục bộ (LSPD) để mở rộng vùng tìm kiếm trên mạng lưới giao thông.

Mặc dù mô hình NS-DBSCAN phản ánh chính xác thực tế hơn so với mô hình Euclid, thách thức lớn nhất của phương pháp này nằm ở chi phí tính toán. Việc xác định vùng lân cận cho hàng nghìn điểm dữ liệu trên các đồ thị phức tạp tạo ra gánh nặng xử lý lớn. Cụ thể, thuật toán gặp phải nút thắt cổ chai về hiệu năng do phải duyệt qua số lượng lớn các cạnh và đỉnh không cần thiết trong quá trình mở rộng mạng lưới, cũng như quy

trình xác định tham số và xử lý nhiễu chưa được tối ưu hóa.

3 Mục tiêu và Giải pháp đề xuất

Nhằm giải quyết bài toán hiệu năng nêu trên, báo cáo này tập trung phân tích và đề xuất các giải pháp cải tiến thuật toán (gọi tắt là iNS-DBSCAN) dựa trên nền tảng nghiên cứu của Wang và cộng sự. Các giải pháp cụ thể bao gồm:

1. Áp dụng chiến lược cắt tỉa cạnh (Edge Pruning) dựa trên định lý tam giác để giảm thiểu không gian tìm kiếm.
2. Tối ưu hóa bảng sắp xếp mật độ nhằm loại bỏ các tính toán dư thừa đối với các điểm dữ liệu thừa thớt.
3. Đơn giản hóa quy trình xác định nhiễu bằng cách tích hợp trực tiếp vào quá trình hình thành cụm.

Mục tiêu cuối cùng của nghiên cứu là xây dựng một phương pháp phân cụm vừa đảm bảo độ chính xác cao đặc trưng của mô hình không gian mạng, vừa đáp ứng được yêu cầu khắt khe về tốc độ xử lý cho các tập dữ liệu quy mô lớn.

Chương II

CÁC KIẾN THỨC CƠ SỞ

Để thực hiện phân cụm dữ liệu trong không gian mạng, việc đầu tiên là cần mô hình hóa hệ thống giao thông thực tế dưới dạng cấu trúc dữ liệu toán học và định nghĩa lại các khái niệm về khoảng cách sao cho phù hợp với môi trường này.

1 Mô hình hóa không gian mạng

Trong báo cáo này, không gian mạng được mô hình hóa bằng một đồ thị vô hướng có trọng số.

Định nghĩa 1: Một đồ thị không gian G được định nghĩa là bộ ba $G = (V, E, W)$, trong đó:

- V : Tập hợp các đỉnh (vertices), đại diện cho các giao lộ.
- E : Tập hợp các cạnh (edges), đại diện cho các đoạn đường nối giữa hai đỉnh.
- W : Tập hợp các trọng số, biểu thị độ dài thực tế của đoạn đường.

Định nghĩa 2: Tập dữ liệu D bao gồm các điểm sự kiện (ví dụ: vị trí tai nạn, điểm dừng xe) nằm tại bất kỳ vị trí nào trên các cạnh E của đồ thị, thay vì chỉ nằm tại các đỉnh như đồ thị truyền thống.

2 Khoảng cách trong Không gian mạng

Sự khác biệt lớn nhất so với các phương pháp cũ nằm ở cách đo khoảng cách.

Định nghĩa 3: Khoảng cách giữa hai điểm p_i và p_j , ký hiệu $dist_N(p_i, p_j)$, là tổng trọng số nhỏ nhất của các cạnh tạo nên đường đi nối hai điểm đó.

$$dist_N(p_i, p_j) = \min \left(\sum_{e \in Path} w(e) \right)$$

Định nghĩa 4: Một đồ thị vô hướng $G = (V, E, W)$, đường đi từ P_0 đến P_n được biểu diễn dưới dạng một chuỗi các đỉnh $P_0, P_1, P_2, \dots, P_{n-1}, P_n$. Đường đi cũng có thể biểu diễn dưới dạng một chuỗi các cạnh: $(P_0, P_1), \dots, (P_{n-1}, P_n) \in E$. Có thể có nhiều đường

đi hoặc không có đường đi nào giữa hai điểm. Khi có nhiều đường đi giữa hai điểm, ta có thể xác định đường đi ngắn nhất.

Định nghĩa 5: Độ dài của đường đi từ điểm P_0 đến điểm P_n đi qua các điểm $P_1, P_2, \dots, P_{n-1}, P_n \in V$ có giá trị là tổng độ dài của tất cả các đoạn của đường đi:

$$d(P_0, P_n) = W(P_0, P_1) + W(P_1, P_2) + \dots + W(P_{n-1}, P_n)$$

Việc sử dụng khoảng cách này giúp đảm bảo tính chính xác thực tế, tránh sai sót khi gom nhóm các điểm gần nhau về tọa độ nhưng bị ngăn cách bởi vật cản.

3 Các định nghĩa Phân cụm

Thuật toán sử dụng hai tham số cơ bản: bán kính Eps và ngưỡng số điểm MinPts.

Định nghĩa 6 (Vùng lân cận mạng lưới): Vùng lân cận của một điểm p , ký hiệu $N_{\text{Eps}}(p)$, là tập hợp tất cả các điểm q sao cho khoảng cách đường đi từ p đến q nhỏ hơn hoặc bằng Eps.

$$N_{\text{Eps}}(p) = \{q \in D \mid \text{dist}_N(p, q) \leq \text{Eps}\}$$

Lưu ý: Vùng lân cận này có hình dạng lan tỏa dọc theo các con đường (dạng mạng nhện) thay vì hình tròn.

Định nghĩa 7: (Điểm lõi) Một điểm p được gọi là Điểm lõi nếu vùng lân cận mạng lưới của nó chứa số lượng điểm dữ liệu lớn hơn hoặc bằng MinPts .

$$|N_{\text{Eps}}(p)| \geq \text{MinPts}$$

Định nghĩa 8: (Điểm biên và nhiễu)

- Điểm biên (Border Point): Là điểm nằm trong vùng lân cận của một điểm lõi nhưng bản thân nó có ít hơn MinPts lân cận.
- Nhiễu (Noise): Là điểm không phải điểm lõi và cũng không phải điểm biên. Đây là các điểm nằm ở khu vực thưa thớt, cách biệt với các cụm dữ liệu chính.

Chương III

ĐỀ XUẤT GIẢI PHÁP TỐI ƯU HÓA THUẬT TOÁN PHÂN CỤM TRONG KHÔNG GIAN MẠNG

Chương này trình bày cơ sở lý thuyết và cơ chế hoạt động của thuật toán phân cụm không gian dựa trên mật độ trong không gian mạng (NS-DBSCAN) nguyên bản, đồng thời phân tích các hạn chế về hiệu năng để dẫn nhập tới các chiến lược cải tiến kỹ thuật (NS-DBSCAND) nhằm tối ưu hóa thời gian thực thi.

1 Cơ chế hoạt động của thuật toán NS-DBSCAN nguyên bản

Thuật toán NS-DBSCAN được đề xuất bởi Wanget al. (2019), là sự mở rộng của thuật toán DBSCAN truyền thống nhằm giải quyết bài toán phân cụm các điểm sự kiện bị ràng buộc bởi mạng lưới giao thông. Khác với DBSCAN sử dụng Euclid trong không gian phẳng, NS-DBSCAN sử dụng khoảng cách đường đi ngắn nhất dọc theo các cạnh của mạng lưới để xác định mức độ tương đồng giữa các đối tượng.

Quy trình vận hành của thuật toán bao gồm hai giai đoạn cốt lõi:

1.1 Giai đoạn 1: Xây dựng trật tự mật độ

Mục tiêu của giai đoạn này là mô hình hóa cấu trúc phân bố mật độ của dữ liệu để hỗ trợ việc lựa chọn tham số đầu vào hợp lý. Giai đoạn này sử dụng tham số bán kính eps và thực hiện qua hai bước con:

- Tính toán lân cận bằng thuật toán LSPD: Thay vì xây dựng ma trận khoảng cách toàn cục tốn kém, thuật toán sử dụng phương pháp mở rộng mạng lưới cục bộ (LSPD). Từ một điểm trung tâm, thuật toán thực hiện mở rộng sang các đỉnh kề. Tại mỗi bước, khoảng cách tích lũy từ tâm được cập nhật ($CDCV$). Quá trình mở rộng sẽ dừng lại khi khoảng cách tích lũy vượt quá ngưỡng eps . Tập hợp các điểm nằm trong phạm vi này cấu thành nên vùng lân cận eps (N_{eps}) của điểm trung tâm.

- Thiết lập bảng và biểu đồ trật tự mật độ: NS-DBSCAN áp dụng chiến lược duyệt đồ thị theo hướng leo đồi để phản ánh cấu trúc cụm tự nhiên. Thuật toán bắt đầu từ một điểm bất kỳ, tìm kiếm trong lân cận điểm có mật độ cao nhất để di chuyển tới, và lặp lại cho đến khi đạt đỉnh mật độ cục bộ. Khi đạt đỉnh, thuật toán ghi nhận các điểm vào Bảng trật tự mật độ theo thứ tự giảm dần của mật độ. Kết quả là các điểm có vị trí không gian gần nhau và mật độ tương đồng sẽ nằm liền kề trong bảng. Dữ liệu này được trực quan hóa thành biểu đồ trật tự mật độ với hình dạng các ngọn đồi, trong đó mỗi ngọn đồi đại diện cho một cụm tiềm năng.

1.2 Giai đoạn 2: Hình thành Cụm

Dựa trên bảng trật tự mật độ, thuật toán tiến hành phân cụm với tham số thứ hai là ngưỡng mật độ tối thiểu ($MinPts$):

- Xác định Điểm Lõi (Core Points): Các điểm trong bảng trật tự mật độ có số lượng lân cận lớn hơn hoặc bằng $MinPts$ được xác định là Điểm Lõi.
- Lan truyền Cụm: Thuật toán khởi tạo cụm mới từ một Điểm Lõi chưa phân loại, sau đó thu nạp toàn bộ các điểm nằm trong vùng lân cận eps của nó (gọi là điểm biên) vào cụm. Quá trình này lặp lại đệ quy: nếu điểm biên mới thêm vào cũng là Điểm Lõi, vùng lân cận của nó tiếp tục được hợp nhất vào cụm hiện tại.
- Xử lý Nhiều: Trong thuật toán gốc, quy trình xác định nhiễu được thực hiện tường minh. Thuật toán kiểm tra từng điểm xem nó có thuộc cụm nào không; nếu không và mật độ thấp hơn $MinPts$, điểm đó được gán nhãn là Nhiều (Noise).

2 Các chiến lược tối ưu hóa trọng thuật toán cải tiến iNS-DBSCAN

Mặc dù NS-DBSCAN giải quyết tốt bài toán phân cụm trên mạng lưới, thuật toán nguyên bản tồn tại các hạn chế về hiệu suất khi xử lý dữ liệu lớn do các thao tác kiểm tra dư thừa. Để khắc phục, phương pháp cải tiến (iNS-DBSCAN) được đề xuất với ba chiến lược tối ưu hóa trọng tâm dựa trên các phân tích lý thuyết và thực nghiệm.

2.1 Chiến lược Lược bỏ Cạnh dựa trên Ràng buộc Độ dài

2.1.1 Phân tích điểm nghẽn của thuật toán gốc

Trong thuật toán gốc (LSPD), để xác định vùng lân cận $N_{eps}(p)$ của một điểm, thuật toán thực hiện mở rộng từ điểm trung tâm sang tất cả các đỉnh kề.

Algorithm1: Local Shortest Path Distance Algorithm

- 1: **Input:** undirected planar graph N , central vertex cp , radius eps
- 2: **Output:** cp 's eps -neighbors $N_{eps}(cp)$

```

3:  $CDCV(cp) = 0$ ,  $CDCV(\text{other vertices}) = \infty$ ,  $cp$  is inserted into a newly initiated queue  $Q$ ;
4: while  $Q$  is not empty do
5:    $p$  is dequeued from  $Q$ ;
6:   if  $p$  is an event vertex and not in  $N_{eps}(p)$  then
7:     add  $p$  to  $N_{eps}(cp)$ ;
8:   end if
9:   for all vertex  $q$  adjacent to  $p$  do
10:    if  $CDCV(p) + W(p, q) < CDCV(q)$  and  $CDCV(p) + W(p, q) \leq eps$  then
11:       $CDCV(q) = CDCV(p) + W(p, q)$ ;
12:      if  $q$  is not in  $Q$  then
13:         $q$  is enqueued to  $Q$ ;
14:      end if
15:    end if
16:  end for
17: end while

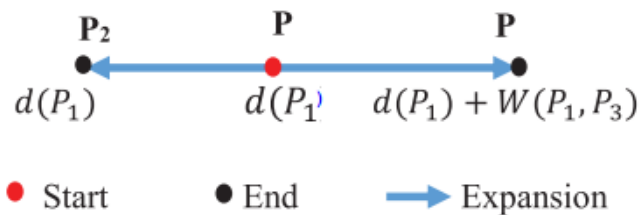
```

Bảng III.1: Pseudocode of local shortest-path distance (LSPD) Algorithm.

- Vấn đề: Tại dòng 8 và 9 của mã giả gốc, thuật toán duyệt qua từng đỉnh kề q và thực hiện phép kiểm tra điều kiện:

$$d(p) + W(p, q) \leq eps$$

Điều này có nghĩa là ngay cả khi một cạnh nối có độ dài rất lớn (ví dụ: 5km) trong



Hình III.1: Minh họa điều kiện

khi bán kính tìm kiếm eps chỉ là 500m, thuật toán vẫn tốn chi phí truy xuất cạnh đó từ bộ nhớ và thực hiện phép toán cộng, so sánh trước khi loại bỏ nó. Trong các mạng lưới giao thông thực tế với hàng nghìn cạnh "dài", sự lãng phí này tích tụ lại là rất lớn.

2.1.2 Cơ chế của thuật toán cải tiến

Chiến lược cải tiến dựa trên một chứng minh toán học chặt chẽ (Định lý 1): "Một đường đi hợp lệ trong lân cận eps không thể chứa bất kỳ cạnh đơn lẻ nào có trọng số lớn hơn eps ".

Từ định lý này, cơ chế tối ưu hóa hoạt động theo hai tầng:

1. Tiền xử lý : Thay vì kiểm tra điều kiện trong vòng lặp (runtime check), thuật toán thực hiện lọc ngay từ đầu vào. Hệ thống rà soát toàn bộ đồ thị và vô hiệu hóa các cạnh có $W(e) > eps$.
 - Việc này làm giảm đáng kể bậc của các đỉnh trong đồ thị tìm kiếm. Khi thuật toán LSPD chạy, nó chỉ nhìn thấy các cạnh tiềm năng, loại bỏ hoàn toàn các thao tác truy xuất bộ nhớ dư thừa cho các cạnh không hợp lệ.
2. Ngắt sớm : Thuật toán cải tiến bổ sung điều kiện dừng chặt chẽ hơn. Nếu khoảng cách tích lũy đến đỉnh hiện tại $d(p)$ đã đạt hoặc vượt quá eps , thuật toán sẽ không tiếp tục xét duyệt các cạnh kề của nó nữa. Điều này ngăn chặn sự lan truyền lãng phí vào các nhánh chắc chắn sẽ vi phạm điều kiện độ dài.

2.2 Tối ưu hóa cấu trúc bảng trật tự mật độ

2.2.1 Phân tích điểm nghẽn trong thuật toán gốc

Giai đoạn xây dựng bảng trật tự mật độ (Algorithm 2) trong bản gốc gặp hai vấn đề lớn về hiệu năng:

1. Chi phí Sắp xếp : Mã giả gốc mô tả việc chèn điểm vào hàng đợi và sau đó "làm cho mật độ các điểm trong Q theo thứ tự từ cao xuống thấp". Việc gọi hàm sắp xếp lại (re-sorting) mỗi khi cập nhật mật độ hoặc thêm điểm mới tạo ra độ phức tạp tính toán lớn, đặc biệt khi số lượng điểm N lớn.
2. Dữ liệu rác : Thuật toán gốc lưu trữ và xử lý tất cả các điểm, kể cả những điểm có mật độ bằng 0 hoặc rất thấp. Những điểm này (Sparse points) về mặt lý thuyết không bao giờ có thể trở thành điểm lõi để tạo cụm, do đó việc tính toán và lưu trữ chúng là lãng phí.

Algorithm2: Generating Density Ordering

- 1: **Input:** undirected planar graph N , radius eps
- 2: **Output:** density ordering table, density ordering graph
- 3: initialize density ordering table;
- 4: **for** each point p **do**
- 5: **if** p is not in density ordering table **then**
- 6: p is inserted into Q and get $N_{eps}(p)$ by LSPD algorithm;
- 7: **end if**
- 8: **while** Q is not empty **do**
- 9: the first point q is dequeued from Q , write the density of q and $N_{eps}(q)$ into the density ordering table;
- 10: **for** each point q in $N_{eps}(q)$ **do**
- 11: **if** q 's density is unknown **then**

```

12:         calculate its density by LSPD algorithm;
13:     end if
14:     if  $q$  is not in density ordering table nor in  $Q$  then
15:         add  $q$  into  $Q$  to make the densities of the points in  $Q$  are from high to
        low.
16:     end if
17: end for
18: else
19:     go to (2);
20: end while
21: end for
22: draw a density ordering graph according to the density ordering table;

```

Bảng III.2: Pseudocode of Generating Density Ordering Table and Graph.

2.2.2 Cơ chế của thuật toán cải tiến

Chiến lược này tối ưu hóa cấu trúc dữ liệu để giảm độ phức tạp từ $O(N \log N)$ xuống thấp hơn:

- Cơ chế chèn bảo toàn:
 - Thay vì thêm vào rồi mới sắp xếp, thuật toán cải tiến tìm vị trí chính xác của điểm q trong hàng đợi Q sao cho thứ tự giảm dần vẫn được bảo toàn và chèn trực tiếp vào đó.
 - Trong khoa học máy tính, việc duy trì một danh sách đã sắp xếp thông qua chèn trực tiếp hiệu quả hơn nhiều so với việc chạy lại thuật toán sắp xếp trên toàn bộ danh sách mỗi lần có thay đổi. Điều này giúp loại bỏ nút thắt cổ chai về thời gian trong vòng lặp chính.
- Lọc bằng ngưỡng Heuristic:
 - Sử dụng ngưỡng $\delta \approx \ln(n)$ làm bộ lọc đầu vào. Điểm có mật độ thấp hơn ngưỡng này bị loại bỏ ngay lập tức.
 - Giải thích: Đây là phương pháp loại suy dựa trên xác suất. Với một tập dữ liệu lớn n , xác suất để một điểm có mật độ cực thấp (ví dụ: $< \ln(n)$) tham gia vào một cụm có ý nghĩa là gần như bằng không. Loại bỏ chúng giúp giảm kích thước N của bài toán, làm cho các vòng lặp phía sau chạy nhanh hơn mà không ảnh hưởng đến độ chính xác của các cụm chính.

2.3 Tích hợp quy trình xác định nhiễu ngầm định

2.3.1 Phân tích điểm nhiễu trong thuật toán gốc

Trong Algorithm 3, thuật toán gốc thực hiện việc xác định nhiễu một cách "thủ công" và tường minh.

Algorithm3: Forming Clusters

```
1: Input: density ordering table, density threshold MinPts
2: Output: density-based clusters

3: for each event point p in density ordering table do
4:   if p does not belong to any cluster or noises then
5:     if the density of p is less than MinPts then
6:       mark p as noise;
7:     else
8:       create a new cluster C containing p;
9:     end if
10:    for each point q in cluster C do
11:      if q is a core point then
12:        for each point s in q's border points do
13:          if s is not a member of cluster C then
14:            add s to cluster C;
15:          end if
16:        end for
17:      end if
18:    end for
19:  end if
20: end for
```

Bảng III.3: Pseudocode of Forming Clusters.

- Vấn đề: Mã giả gốc chứa các dòng lệnh kiểm tra điều kiện lồng nhau:
 - Dòng 4: if *p* does not belong to any cluster or noises then
 - Dòng 5: if density of *p* < *MinPts* then
 - Dòng 6: mark *p* as noise Việc lặp đi lặp lại các phép kiểm tra if-else này cho hàng nghìn điểm dữ liệu, đặc biệt là việc phải truy xuất trạng thái "noise" của từng điểm, làm tăng độ trễ thực thi.

2.3.2 Cơ chế của thuật toán cải tiến

Thuật toán cải tiến chuyển đổi tư duy từ tìm nhiễu sang định nghĩa nhiễu bằng phương pháp loại trừ.

- Loại bỏ hoàn toàn các dòng lệnh 3 và 4, và đơn giản hóa dòng lệnh 2 trong mã giả gốc. Thuật toán chỉ tập trung duy nhất vào nhiệm vụ: Tìm và Mở rộng Cụm.
- Giải thích: Trong logic tập hợp, ~~Nhiều (Noise) chính là phần bù của hợp các Cụm (Clusters):~~

$$Noise = Data \setminus \bigcup Clusters$$

Thay vì tốn tài nguyên CPU để kiểm tra và gán nhãn cho từng điểm nhiễu trong vòng lặp, thuật toán cải tiến để mặc định trạng thái của các điểm là chưa phân loại. Sau khi thuật toán kết thúc, tất cả các điểm đã được gom vào các cụm; những điểm còn sót lại thì tự động được hiểu là Nhiều mà không cần một bước xử lý hay kiểm tra điều kiện nào cả. Chiến lược làm một việc này giúp mã nguồn gọn nhẹ và tốc độ xử lý nhanh hơn đáng kể.

3 Mã giả và quy trình thực thi thuật toán cải tiến

Dựa trên ba chiến lược tối ưu hóa đã phân tích, quy trình tổng thể của thuật toán iNS-DBSCAN được hình thức hóa thông qua mã giả trong các hình.

Thuật toán 1. Khoảng cách đường đi ngắn nhất cục bộ (LSPD)

- Đầu vào: Đồ thị phẳng vô hướng N , đỉnh trung tâm cp , bán kính eps .
- Đầu ra: Vùng lân cận của đỉnh trung tâm $N_{Eps}(cp)$ (Tập hợp các đỉnh lân cận của đỉnh cp).

Algorithm 1 Improved Local Shortest-Path Distance (LSPD)

1: **Input:**

2: $G = (V, E, W)$: Undirected weighted graph

3: cp : Central point ($cp \in V$)

4: ϵ : Radius threshold

5: **Output:**

6: $N_\epsilon(cp)$: Set of neighbors within distance ϵ

7: **Preprocessing:** Remove all edges $e \in E$ where $W(e) > \epsilon$.

8: Initialize $d[v] \leftarrow \infty$ for all $v \in V$

9: $d[cp] \leftarrow 0$

10: $Q \leftarrow \{cp\}$

▷ Queue for expansion

11: $N_\epsilon(cp) \leftarrow \emptyset$

12: **while** $Q \neq \emptyset$ **do**

13: $p \leftarrow Q.dequeue()$

14: **if** $p \notin N_\epsilon(cp) \wedge p \neq cp$ **then**

```

15:      $N_\epsilon(cp) \leftarrow N_\epsilon(cp) \cup \{p\}$ 
16: end if
17:      $\triangleright$  Improvement: Stop expansion if limit reached
18: if  $d[p] < \epsilon$  then
19:     for all vertex  $q$  adjacent to  $p$  do
20:          $new\_dist \leftarrow d[p] + W(p, q)$ 
21:          $\triangleright$  Check distance constraint
22:         if  $new\_dist < d[q] \wedge new\_dist \leq \epsilon$  then
23:              $d[q] \leftarrow new\_dist$ 
24:             if  $q \notin Q$  then
25:                  $Q.enqueue(q)$ 
26:             end if
27:         end if
28:     end for
29: end if
30: end while
31: return  $N_\epsilon(cp)$ 

```

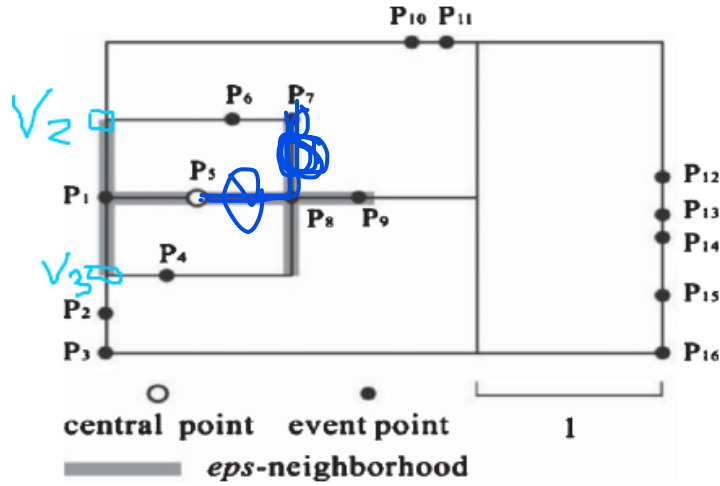
Hình III.2: Mô tả thuật toán LSPD cải tiến

Thuật toán gốc: Dòng 8 cho thấy thuật toán vẫn tiếp tục kiểm tra để thực hiện mở rộng cơ bản khi khoảng cách đã đạt đến ϵ .

Cải tiến: Khi tổ chức dữ liệu, chúng tôi loại bỏ các cạnh có độ dài vượt quá ϵ vì đường đi có độ dài tối đa là ϵ chắc chắn không chứa các cạnh vượt quá ϵ . Điều đó giúp giảm số lượng điểm liên kề của vòng lặp trong dòng 7. Tại điểm mà độ dài từ điểm trung tâm đến nó đã đạt đến khoảng cách ϵ , chúng tôi không kiểm tra để mở rộng nữa: chúng tôi dừng thao tác kiểm tra sau khi các đỉnh đã đạt đến ϵ .

Minh họa: Trong Hình III.2, giả sử $\epsilon = 1$, điểm trung tâm được chọn ngẫu nhiên là P_5 . Theo Thuật toán 1, vùng lân cận trong khoảng cách ϵ của P_5 ($N_\epsilon(P_5)$) như sau:

- Mở rộng cơ bản từ P_5 đến các đỉnh P_1 và P_8 , tính d cho P_1 và P_8 :
 - $d(P_1) = d(P_5) + W(P_5, P_1) = 0 + 0.5 = 0.5$
 - $d(P_8) = d(P_5) + W(P_5, P_8) = 0 + 0.5 = 0.5$
- Các giá trị mới của $d(P_1)$ và $d(P_8)$ nhỏ hơn d hiện tại của chúng và nhỏ hơn ϵ , vì vậy P_5 được mở rộng đến P_1 và P_8 , nghĩa là P_1 và P_8 trở thành đỉnh bắt đầu trong lần mở rộng tiếp theo.
- Đối với P_1 : Tiếp tục mở rộng cơ bản từ P_1 đến V_2 và P_1 đến V_3 , ta có:



Hình III.3: Minh họa mở rộng với Central point: P5

- $d(V_2) = d(P_1) + W(P_1, V_2) = 0.5 + 0.5 = 1 = eps$
- $d(V_3) = d(P_1) + W(P_1, V_3) = 0.5 + 0.5 = 1 = eps$
- Đối với P_8 : Mở rộng cơ bản từ P_8 đến P_4 , P_8 đến P_7 và P_8 đến P_9 .
 - $d(P_4) = d(P_8) + W(P_8, P_4) = 1 = eps$
 - $d(P_7) = d(P_8) + W(P_8, P_7) = 1 = eps$
 - $d(P_9) = d(P_8) + W(P_8, P_9) = 1 = eps$
- Các điểm V_2, V_3, P_4, P_7, P_9 đã đạt đến eps nên chúng tôi không mở rộng nữa và kết luận có thể là: $N_{eps}(P_5) = \{P_1, P_8, P_7, P_9, V_2, V_3, P_4\}$. Trong khi thuật toán gốc vẫn tiếp tục mở rộng: "Việc mở rộng tiếp tục với V_2, V_3, P_7, P_9 và P_4 là các đỉnh bắt đầu cho đến khi tất cả các đường mở rộng bị chặn".

Thuật toán 2. Tạo sắp xếp mật độ

- Đầu vào: Đồ thị phẳng vô hướng N , bán kính eps
- Đầu ra: Bảng sắp xếp mật độ, biểu đồ sắp xếp mật độ

Algorithm 2 Improved Generating Density Ordering

1: **Input:**

- 2: G : Network graph
- 3: ϵ : Radius parameter
- 4: n : Total number of points

5: **Output:**

- 6: T : Density ordering table

7: Initialize $T \leftarrow \emptyset$

8: $\delta \leftarrow \ln(n)$

▷ Heuristic threshold for low density

```

9: Initialize priority queue  $Q$ 
10: for all point  $p \in G$  do
11:   if  $p \notin T$  then
12:     Calculate  $N_\epsilon(p)$  and  $Density(p)$  using Algorithm 1
13:      $Q.insert(p)$ 
14:     while  $Q \neq \emptyset$  do
15:        $q \leftarrow Q.dequeue()$  ▷ Point with highest density
16:       ▷ Improvement: Filter out low-density points
17:       if  $Density(q) \geq \delta$  then
18:         Write entry  $(q, Density(q), N_\epsilon(q))$  into  $T$ 
19:         for all  $r \in N_\epsilon(q)$  do
20:           if  $r \notin T \wedge r \notin Q$  then
21:             if  $Density(r)$  is unknown then
22:               Calculate  $N_\epsilon(r)$  and  $Density(r)$  via Algorithm 1
23:             end if
24:             ▷ Improvement: Insert maintaining descending order
25:              $Q.insert\_descending(r)$ 
26:           end if
27:         end for
28:       end if
29:     end while
30:   end if
31: end for
32: return  $T$ 

```

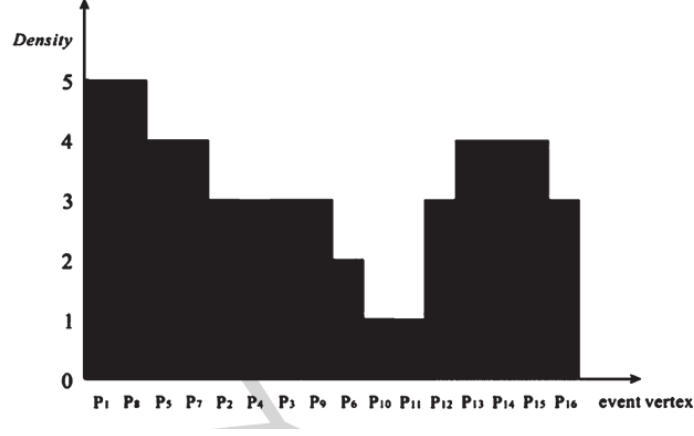
Hình III.4: Thuật toán sắp xếp mật độ cải tiến (Improved Generating Density Ordering)

Thuật toán gốc: Tại dòng 15 và trong minh họa thuật toán của [3], thuật toán bắt đầu sắp xếp Q sau khi dữ liệu được chèn vào Q . Bảng sắp xếp mật độ được xây dựng chứa tất cả các điểm, bao gồm cả những điểm có mật độ bằng 0 và không có hàng xóm nào cả. Những điểm này chắc chắn không thuộc về bất kỳ cụm nào.

Cải tiến: Khi chèn một điểm vào hàng đợi Q , chúng ta nên chèn điểm đó theo thứ tự giảm dần của giá trị mật độ để duy trì thứ tự ban đầu của mảng, nhờ đó Q không cần phải được sắp xếp lại. Vì việc đưa các đỉnh vào Q đã đúng theo thứ tự sắp xếp, nên kết quả thu được là chính xác.

Bản gốc mô tả chung chung là thêm vào hàng đợi rồi sắp xếp lại hoặc mô tả việc duy trì thứ tự nhưng không nhấn mạnh kỹ thuật chèn. Trong bản cải tiến sử dụng kỹ thuật chèn trực tiếp vào vị trí đúng giúp duy trì tính chất giảm dần của hàng đợi mà không cần gọi hàm sort lại toàn bộ.

Minh họa: Một bảng T trống được khởi tạo. Theo thứ tự tên của các điểm sự kiện, P_1



Hình III.5: Bảng sắp xếp mật độ

được chọn trước, và tính toán $N_eps(P_1) = \{P_5, P_2, P_4, P_8, P_3\}$ và hàng đợi $Q = \{P_1\}$. Q không trống. Lấy P_1 từ Q và đưa bản ghi (ID: P_1 ; density: 5; $N_eps : \{P_5, P_2, P_4, P_8, P_3\}$) vào bảng sắp xếp mật độ T .

Mật độ của các điểm trong tập hợp $N_eps(P_1)$ được tính toán và đưa vào vị trí thích hợp của Q sao cho Q có thứ tự mật độ từ cao xuống thấp:

- $N_eps(P_5) = (P_1, P_8, P_9, P_7) \rightarrow Q = \{P_5\}$
- $N_eps(P_2) = (P_3, P_4, P_1) \rightarrow Q = \{P_5, P_2\}$
- $N_eps(P_4) = (P_2, P_1, P_3) \rightarrow Q = \{P_5, P_4, P_2\}$
- $N_eps(P_8) = (P_5, P_9, P_1, P_7, P_6) \rightarrow Q = \{P_8, P_5, P_4, P_2\}$
- $N_eps(P_3) = (P_2, P_4, P_1) \rightarrow Q = \{P_8, P_5, P_3, P_4, P_2\}$

$Q = \{P_8, P_5, P_3, P_4, P_2\}$ với mật độ tương ứng theo thứ tự giảm dần là 5, 4, 3, 3 và 3. Theo thuật toán gốc, nếu các điểm trong $N_eps(P_1)$ được đưa vào tuân theo thứ tự đã sử dụng trong $N_eps(P_1)$, sự sắp xếp của các điểm đó sẽ là $Q = \{P_5, P_2, P_4, P_8, P_3\}$. Mặt khác, nếu các điểm trong Q được sắp xếp theo thứ tự mật độ của chúng từ cao xuống thấp, đó là $Q = \{P_8, P_5, P_2, P_4, P_3\}$ như trong [3]. Sau đó, lấy ra điểm đầu tiên có mật độ lớn hơn hoặc bằng 3 trong Q (P_8) và đặt nó vào bảng sắp xếp mật độ T .

$$T = \{(P_1, 5, \{P_5, P_2, P_4, P_8, P_3\}), (P_8, 5, \{P_5, P_9, P_1, P_7, P_6\})\}$$

Các giá trị mật độ của các điểm P_7 , P_9 và P_6 được tính lần lượt là 4, 3 và 3. Hàng đợi Q được cập nhật là $\{P_5, P_7, P_2, P_4, P_3, P_9, P_6\}$. Sau đó, điểm đầu tiên trong Q được loại bỏ khỏi hàng đợi để tiếp tục thực hiện các thao tác nêu trên cho đến khi Q trống. Kết quả là, thứ tự của các điểm trong T là $P_1, P_8, P_5, P_7, P_2, P_4, P_3, P_9$. Đây là các điểm liên quan đến P_1 về quan hệ lân cận cho khoảng cách eps .

Tiếp tục xem xét các điểm từ P_2 đến P_{16} . Các điểm từ P_2 đến P_9 đã có trong bảng sắp xếp mật độ.

Tương tự như P_1 , ta xem xét P_{10} : $Q = \{P_{10}\}$, $N_eps(P_{10}) = \{P_{11}\}$. P_{10} không được bao gồm trong bảng sắp xếp mật độ T vì $Density(P_{10}) = 1$. Tương tự, $Density(P_{11}) = 1$ nên P_{11} không tồn tại trong T.

Tiếp tục với P_{12} : $Q = \{P_{12}\}$, $N_eps(P_{12}) = \{P_{13}, P_{14}, P_{15}\}$. $Density(P_{12}) = 3$. Thêm P_{12} vào bảng sắp xếp mật độ T bao gồm các điểm $P_1, P_8, P_5, P_7, P_2, P_4, P_3, P_9$ và P_{12} .

Làm tương tự như với P_1 , chúng ta nhận được kết quả T như sau:

$$T = \{(P_1, 5, \{P_5, P_2, P_4, P_8, P_3\}), \\ (P_8, 5, \{P_5, P_9, P_1, P_7, P_6\}), \\ (P_5, 4, \{P_1, P_8, P_9, P_7\}), \\ (P_7, 4, \{P_8, P_6, P_5, P_9\}), \\ (P_2, 3, \{P_3, P_4, P_1\}), \\ (P_4, 3, \{P_2, P_1, P_3\}), \\ (P_3, 3, \{P_2, P_4, P_1\}), \\ (P_9, 3, \{P_8, P_5, P_7\}), \\ (P_6, 2, \{P_7, P_8\}), \\ (P_{12}, 3, \{P_{13}, P_{14}, P_{15}\}), \\ (P_{13}, 4, \{P_{12}, P_{14}, P_{15}, P_{16}\}), \\ (P_{14}, 4, \{P_{13}, P_{12}, P_{15}, P_{16}\}), \\ (P_{15}, 4, \{P_{12}, P_{13}, P_{14}, P_{16}\}), \\ (P_{16}, 3, \{P_{13}, P_{14}, P_{15}\})\}$$

So với thuật toán gốc, có cả P_{10} và P_{11} trước P_{12} . Phải mất một khoảng thời gian để kiểm tra các điểm này trong bước tiếp theo là hình thành các cụm. Biểu đồ sắp xếp mật độ của mạng lưới đường được mô tả trong Hình III.5.

Thuật toán 3. Hình thành các cụm

Logic xây các cụm: Khởi tạo cụm với các điểm lõi và dần dần thêm các điểm biên vào cụm cho đến khi không còn điểm biên nào được thêm vào.

- Đầu vào: Bảng sắp xếp mật độ, ngưỡng mật độ $MinPts$
- Đầu ra: Các cụm dựa trên mật độ

Algorithm 3 Improved Cluster Formation

- 1: **Input:**
- 2: T : Density ordering table
- 3: $MinPts$: Minimum density threshold
- 4: **Output:**

```

5:  Clusters: Set of formed clusters

6:  Clusters  $\leftarrow \emptyset$ 
7:  for all point  $p \in T$  do
8:       $\triangleright$  Improvement: Skip explicit noise check loops
9:      if  $p$  is not classified in any cluster then
10:         if  $Density(p) \geq MinPts$  then
11:             Create new cluster  $C$ 
12:              $C \leftarrow C \cup \{p\}$ 
13:              $\triangleright$  Expand cluster using border points
14:             for all point  $q \in C$  do  $\triangleright$  Iterate dynamically over growing cluster
15:                 if  $q$  is a Core Point ( $Density(q) \geq MinPts$ ) then
16:                     for all neighbor  $s \in N_\epsilon(q)$  do
17:                         if  $s \notin C$  then
18:                              $C \leftarrow C \cup \{s\}$ 
19:                         end if
20:                     end for
21:                 end if
22:             end for
23:              $Clusters \leftarrow Clusters \cup \{C\}$ 
24:         end if
25:     end if
26: end for
27:  $\triangleright$  Note: Points not in any  $C \in Clusters$  are implicitly Noise
28: return Clusters

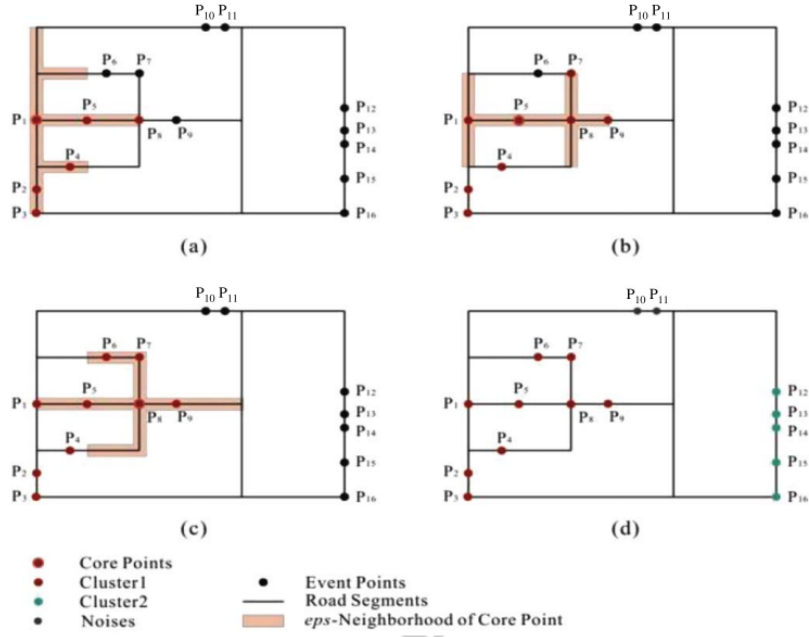
```

Hình III.6: Thuật toán hình thành cụm cải tiến (Improved Cluster Formation)

Thuật toán gốc: Tại dòng 4, thuật toán kiểm tra xem các điểm sự kiện có thuộc về bất kỳ cụm hoặc tập hợp nhiễu nào hay không, dòng 5 và 6 sau đó tiếp tục đánh giá điều kiện của các điểm có mật độ nhỏ hơn $MinPts$ để gán chúng cho nhiễu. Quá trình đánh giá xem các điểm có thuộc về bất kỳ cụm hoặc tập hợp nhiễu nào trong vòng lặp for của dòng 1 gây tốn thời gian. Hơn nữa, có thể gán sai các điểm biên có mật độ nhỏ hơn $MinPts$ cho nhiễu.

Cải tiến: Nhiễu không được xác định trong quá trình hình thành cụm. Nhiễu là những điểm không thuộc về bất kỳ cụm nào. Tức là, các dòng 5 và 6 không cần phải thực hiện để xác định nhiễu. Hơn nữa, câu lệnh trong dòng 4 cũng bỏ qua việc kiểm tra xem một điểm sự kiện có thuộc về tập hợp nhiễu hay không. Điều này giúp loại bỏ ba thao tác kiểm tra trong vòng lặp của dòng 3. Do đó, thời gian thực thi thuật toán giảm xuống.

Minh họa: Xem xét $eps = 1$, $MinPts = 4$. P_1 được chọn làm điểm lõi và P_1 không thuộc cụm nào. Do đó, một cụm mới, cụ thể là C_1 , được tạo ra chứa P_1 . Các điểm biên của P_1



Hình III.7: Minh họa

$(N_eps(P_1) = \{P_5, P_2, P_4, P_8, P_3\})$ được thêm vào C_1 nên $C_1 = \{P_1, P_5, P_2, P_4, P_8, P_3\}$. Điểm lõi được chọn tiếp theo trong C_1 là P_5 . Tập hợp các điểm biên của P_5 là $\{P_1, P_8, P_7, P_9\}$, trong đó P_1 và $P_8 \in C_1$, P_7 và $P_9 \notin C_1$, nên P_7 và P_9 được thêm vào C_1 (Hình III.4.b), $C_1 = \{P_1, P_5, P_2, P_4, P_8, P_3, P_9, P_7\}$. Mật độ của P_2 và P_4 là 3, nhỏ hơn giá trị của MinPts (4), nên không có thêm điểm nào được thêm vào C_1 . Tương tự, P_8 là điểm lõi, nên các điểm biên $\{P_5, P_9, P_1, P_7, P_6\}$ được thêm vào C_1 (Hình 6c): $C_1 = \{P_1, P_5, P_2, P_4, P_8, P_3, P_9, P_7, P_6\}$. P_3, P_9 và P_6 không phải là điểm lõi. P_7 là điểm lõi nhưng tất cả các điểm biên của nó là $\{P_8, P_6, P_5, P_9\}$ đều thuộc về C_1 . Cuối cùng, cụm $C_1 = \{P_1, P_5, P_2, P_4, P_8, P_3, P_7, P_9, P_6\}$ được hình thành. Tương tự, thuật toán gán P_{12} đến P_{16} cho cụm mới C_2 . Kết quả của quá trình phân cụm được hiển thị trong Hình 6d. Thuật toán gốc có khả năng phát hiện nhiễu, và do đó nó đánh dấu P_{10} và P_{11} là nhiễu.

Chương IV

Đánh giá và So sánh Hiệu năng Thực thi

1 Thiết lập thực nghiệm và Phương pháp đánh giá

Để đảm bảo tính khách quan và khả năng ứng dụng thực tiễn của nghiên cứu, dữ liệu đầu vào phục vụ cho quá trình thực nghiệm được trích xuất trực tiếp từ hệ thống OpenStreetMap (OSM). Việc sử dụng dữ liệu địa lý thực tế giúp phản ánh chính xác sự phân bố phức tạp và tính ngẫu nhiên của các đối tượng không gian, tạo ra môi trường kiểm thử sát với thực tế nhất.

Quy trình thực nghiệm được thiết kế để đo lường hiệu năng xử lý song song giữa hai phiên bản: Thuật toán gốc (Original Algorithm) và Thuật toán cải tiến (Improved Algorithm). Các kịch bản kiểm thử được xây dựng dựa trên sự biến thiên của không gian tham số đầu vào, bao gồm bán kính lân cận (ϵ - Eps) và ngưỡng mật độ điểm tối thiểu ($MinPts$).

Chúng tôi tập trung phân tích hai chỉ số định lượng chính:

1. Thời gian thực thi (Time): Tổng thời gian xử lý của CPU (đơn vị: ms) để hoàn tất phân cụm.
2. Mức độ chênh lệch hiệu năng (Diff %): Chỉ số phần trăm thể hiện sự tương quan tốc độ giữa hai thuật toán, được tính theo công thức:

$$Diff(\%) = \frac{Time_{Original} - Time_{Improved}}{Time_{Original}} \times 100$$

- Nếu $Diff > 0$: Thuật toán cải tiến chạy nhanh hơn.
- Nếu $Diff < 0$: Thuật toán cải tiến chạy chậm hơn (thường do chi phí khởi tạo cấu trúc dữ liệu phụ trợ trong các tác vụ nhẹ).

Kết quả thực nghiệm chi tiết được tổng hợp tại Bảng 4.1 dưới đây.

Eps	MinPts	Original(ms)	Improved(ms)	Diff(%)
200	20	113.83	92.00	19.17
200	25	72.10	72.62	-0.72
200	30	75.43	73.73	2.25
200	35	69.25	67.07	3.14
200	40	112.05	113.79	-1.56
250	20	139.05	142.25	-2.30
250	25	134.72	137.71	-2.22
250	30	198.92	202.73	-1.92
250	35	132.08	129.77	1.74
250	40	119.08	118.78	0.25
300	20	243.25	232.19	4.55
300	25	219.63	218.45	0.54
300	30	304.53	301.07	1.14
300	35	198.70	198.35	0.18
300	40	193.29	199.05	-2.98
350	20	356.33	354.71	0.46
350	25	353.20	356.42	-0.91
350	30	332.89	331.12	0.53
350	35	398.39	322.49	19.05
350	40	300.20	307.19	-2.33
400	20	623.17	531.41	14.73
400	25	524.54	527.00	-0.47
400	30	635.12	597.40	5.94
400	35	514.78	537.78	-4.47
400	40	544.35	484.64	10.97

Hình IV.1: So sánh thuật toán gốc và cải tiến – Dữ liệu Cần Thơ

2 Phân tích và Nhận xét kết quả

Dựa trên số liệu thực nghiệm từ Bảng 4.1, chúng tôi đưa ra những phân tích sâu về hành vi của thuật toán cải tiến như sau:

Thứ nhất, hiệu quả vượt trội trong các tác vụ phức tạp: Quan sát bảng dữ liệu, ta thấy thuật toán cải tiến phát huy hiệu quả rõ rệt nhất ở những kịch bản đòi hỏi khối lượng tính toán lớn hoặc khi tham số đầu vào tạo ra không gian tìm kiếm rộng.

- Điển hình tại trường hợp Eps=200, MinPts=20, thuật toán cải tiến giảm thời gian từ 113.83ms xuống còn 92.00ms, đạt mức cải thiện hiệu năng cao nhất là 19.17
- Tương tự, tại các ngưỡng xử lý nặng với Eps=400, nơi thời gian thực thi của thuật toán gốc vượt quá 600ms, thuật toán cải tiến vẫn duy trì sự ổn định tốt với mức cải thiện đáng kể (đạt 14.73% tại MinPts=20 và 10.97

Thứ hai, đánh giá về sự dao động và chi phí đánh đổi (Trade-off): Trong một số trường hợp cụ thể (ví dụ: Eps=200/MinPts=25 hoặc Eps=400/MinPts=35), chỉ số Diff mang giá trị âm nhẹ (từ -0.47

- Nguyên nhân: Điều này có thể được giải thích do chi phí khởi tạo (overhead) của các cấu trúc dữ liệu tối ưu hóa (như Grid hoặc Index) lấn át lợi ích thu được khi dữ liệu phân bố thưa hoặc số lượng phép tính toán khoảng cách không đủ lớn.
- Tuy nhiên, mức sụt giảm này là không đáng kể (thường chỉ vài mili-giây) so với lợi ích to lớn về mặt thời gian đạt được ở các trường hợp dữ liệu dày đặc (tiết kiệm được gần 100ms tại Eps=400/MinPts=20).

Thứ ba, kết luận chung: Tổng quan, thuật toán cải tiến cho thấy khả năng chịu tải tốt hơn (better scalability). Khi độ phức tạp của bài toán tăng lên (Eps lớn), xu hướng cải thiện hiệu năng trở nên rõ ràng và tích cực hơn. Điều này khẳng định tính đúng đắn của phương pháp đề xuất khi áp dụng cho các bài toán phân cụm dữ liệu OpenStreetMap quy mô lớn, nơi ưu tiên rút ngắn thời gian xử lý các tác vụ nặng hơn là tối ưu vi mô cho các tác vụ nhẹ.

Chương V

Kết luận và Hướng phát triển

1 Kết luận

Nghiên cứu này đã tập trung giải quyết bài toán tối ưu hóa hiệu năng cho thuật toán phân cụm dữ liệu không gian, với mục tiêu khắc phục các hạn chế về thời gian thực thi của thuật toán gốc khi áp dụng trên các tập dữ liệu lớn. Thông qua việc đề xuất và cài đặt giải thuật cải tiến, kết hợp với quy trình kiểm thử thực nghiệm trên dữ liệu thực tế từ OpenStreetMap, chúng tôi rút ra các kết luận chính sau:

- Về tính hiệu quả: Giải thuật cải tiến đã chứng minh được sự ưu việt về tốc độ xử lý trong đa số các kịch bản thử nghiệm. Đặc biệt, tại các ngưỡng tham số đòi hỏi khối lượng tính toán lớn (ví dụ: *Eps* lớn và *MinPts* nhỏ), thuật toán đề xuất đạt mức cải thiện hiệu năng ấn tượng (lên đến xấp xỉ 20%), khẳng định khả năng xử lý tốt các vùng dữ liệu có mật độ phức tạp.
- Về khả năng mở rộng (Scalability): Kết quả đối sánh cho thấy thuật toán cải tiến có độ ổn định cao hơn. Mặc dù tồn tại sự đánh đổi nhỏ (trade-off) về chi phí khởi tạo trong các tác vụ nhẹ, nhưng xu hướng chung cho thấy thuật toán càng phát huy tác dụng khi quy mô và độ phức tạp của bài toán tăng lên.
- Về tính thực tiễn: Việc thử nghiệm thành công trên dữ liệu bản đồ số OSM khẳng định tiềm năng ứng dụng của thuật toán trong các hệ thống thông tin địa lý (GIS), phân tích giao thông và các dịch vụ dựa trên vị trí (LBS), nơi yêu cầu độ trễ thấp là yếu tố then chốt.

2 Hướng phát triển

Dựa trên những kết quả đã đạt được và những hạn chế còn tồn tại, chúng tôi đề xuất một số hướng nghiên cứu và phát triển tiếp theo như sau:

1. Tối ưu hóa bộ nhớ: Hiện tại nghiên cứu tập trung sâu vào tối ưu thời gian CPU. Hướng tiếp theo sẽ xem xét việc tối ưu hóa dung lượng bộ nhớ tiêu thụ, đặc biệt là việc tinh gọn các cấu trúc dữ liệu phụ trợ được sử dụng trong quá trình đánh chỉ mục không gian.

2. Xử lý song song và phân tán (Parallel and Distributed Processing): Tận dụng sức mạnh của các kiến trúc đa luồng (Multi-threading) hoặc tính toán trên GPU để song song hóa quá trình tìm kiếm lân cận, nhằm xử lý các tập dữ liệu quy mô cực lớn (Big Data) mà một máy đơn lẻ khó đáp ứng.
3. Thử nghiệm trên các miền dữ liệu khác: Mở rộng phạm vi đánh giá thuật toán trên các loại dữ liệu không gian đa chiều khác ngoài dữ liệu bản đồ 2D (ví dụ: dữ liệu 3D Point Cloud trong quét laser hoặc dữ liệu y sinh).
4. Cơ chế tham số thích nghi (Adaptive Parameters): Nghiên cứu tích hợp các kỹ thuật học máy để tự động gợi ý hoặc điều chỉnh các tham số Eps và $MinPts$ tối ưu dựa trên đặc trưng phân bố của từng vùng dữ liệu, giúp thuật toán linh hoạt hơn mà không cần can thiệp thủ công quá nhiều.

Chương VI

LỜI CẢM ƠN

Để hoàn thành bài tập lớn này, bên cạnh sự nỗ lực và phối hợp của các thành viên trong nhóm, chúng em đã nhận được rất nhiều sự quan tâm, hướng dẫn quý báu.

Trước hết, nhóm chúng em xin gửi lời cảm ơn chân thành đến Nguyễn Thủy Đoan Trang. Cô đã tận tình giảng dạy, trang bị nền tảng kiến thức môn học và đưa ra những định hướng cụ thể giúp nhóm có thể tiếp cận vấn đề và triển khai bài tập một cách hiệu quả nhất.

Chúng em cũng xin gửi lời cảm ơn đến các Thầy Cô trong khoa Công nghệ Thông tin, Trường Đại học Nha Trang đã tạo điều kiện thuận lợi về môi trường học tập và tài liệu tham khảo trong suốt quá trình chúng em theo học tại trường.

Đồng thời, nhóm cũng xin cảm ơn cộng đồng OpenStreetMap và các tác giả của những nghiên cứu liên quan đã cung cấp nguồn dữ liệu và cơ sở lý thuyết quan trọng, giúp chúng em có tư liệu để thực hiện phần thực nghiệm cho bài báo cáo này.

Mặc dù cả nhóm đã rất cố gắng, nhưng do thời gian thực hiện và kinh nghiệm thực tiễn còn hạn chế, bài tập lớn chắc chắn không tránh khỏi những thiếu sót. Chúng em rất mong nhận được những lời nhận xét, góp ý thẳng thắn từ Cô để bài làm được hoàn thiện hơn, cũng như để chúng em rút ra được những bài học kinh nghiệm quý báu cho các dự án sau này.

Chúng em xin chân thành cảm ơn!

TÀI LIỆU THAM KHẢO

- [1] Phạm Văn Nam, *Lập trình Python*
- [2] Đoàn Vũ Thịnh, *Slide Bài giảng Trí tuệ nhân tạo*
- [3] Tianfu Wang, Chang Ren, Yun Luo, JingTian - *NS-DBSCAN: A Density-Based Clustering Algorithm in Network Space* (2019)
- [4] Trang T.D.Nguyen, Loan.T.T.Nguyen, Anh Nguyen, Unil Yun, Bay Vo - *A method for efficient clustering of spatial data in network space*
- [5] Kilian Weinberger, *The DBSCAN Clustering Algorithm Explained*. https://www.youtube.com/watch?v=4AW_5nYQkuc.
- [6] MÌ AI, *Phân cụm dữ liệu dựa trên mật độ không gian với thuật toán DBSCAN - MÌ AI*. https://www.youtube.com/watch?v=_cLje7QCCYU.