



Tối ưu hóa Phân cụm Không gian trên Mạng lưới Đô thị

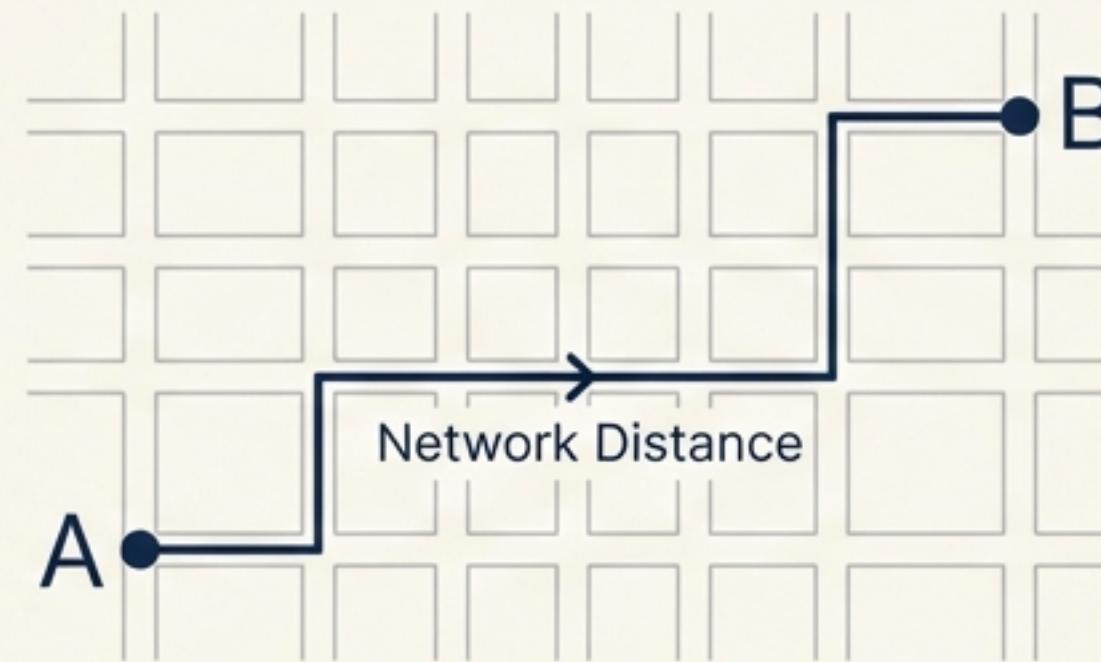
- Sự bùng nổ của dữ liệu không gian lớn (Spatial Big Data)
- Yêu cầu cấp thiết về các thuật toán phân tích hiệu quả
- Ứng dụng trong: quy hoạch đô thị, phân tích kinh tế, quản lý tài nguyên.

NS-DBSCAN: Một giải pháp tiềm năng cho không gian mạng lưới

Khoảng cách Euclidean

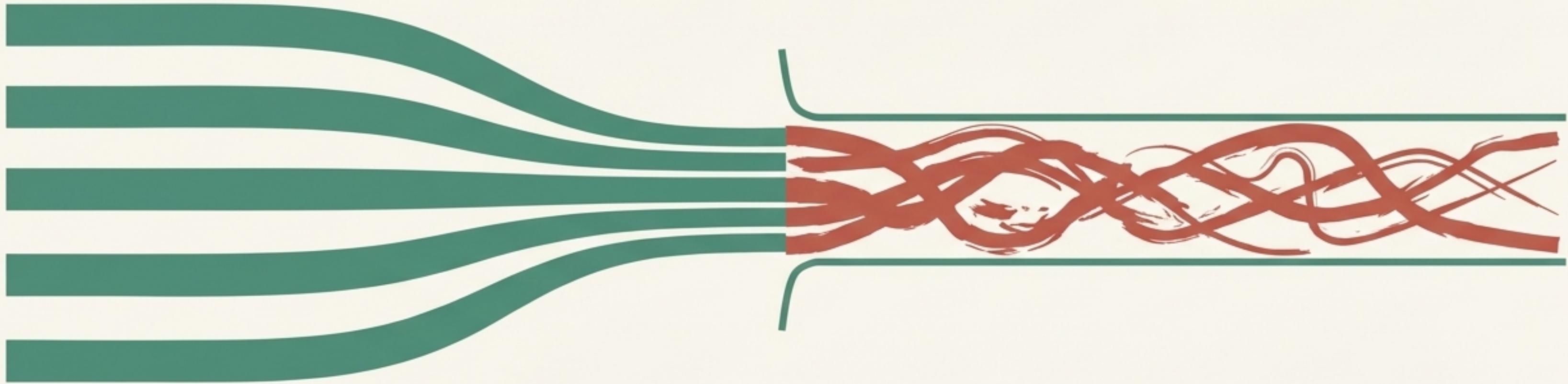


Khoảng cách Mạng lưới



- Dựa trên thuật toán DBSCAN nổi tiếng.
- Thích ứng cho không gian mạng lưới (đường đi, sông ngòi).
- Sử dụng khoảng cách đường đi ngắn nhất cục bộ (LSPD).
- **Vấn đề cốt lõi:** Quá trình phân cụm tốn nhiều thời gian, đặc biệt với các tập dữ liệu lớn.

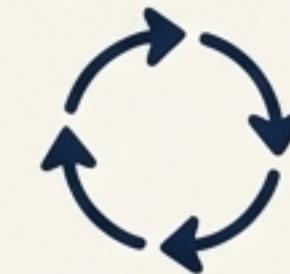
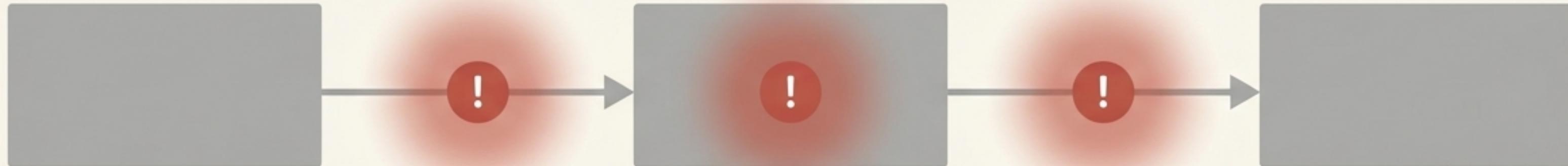
Điểm nghẽn hiệu năng của NS-DBSCAN



Tại sao thuật toán lại chậm?

- Các thao tác tính toán dư thừa.
- Xử lý các điểm không cần thiết.
- Lãng phí tài nguyên và thời gian, đặc biệt khi dữ liệu lớn.

Phân tích các điểm kém hiệu quả chính



Thuật toán LSPD

- Vẫn kiểm tra các cạnh có độ dài lớn hơn ngưỡng `eps`.
- Thao tác không cần thiết.

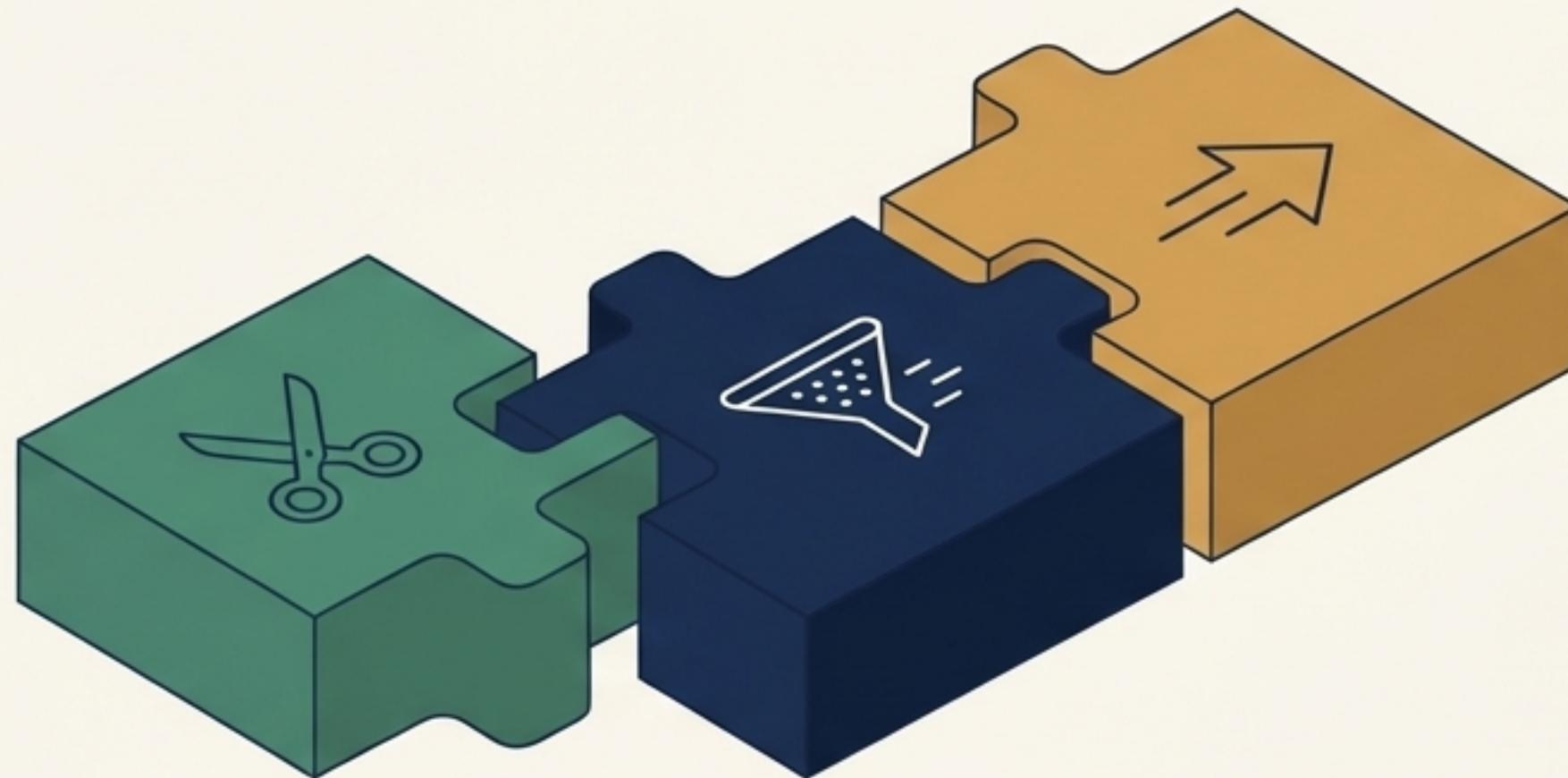
Bảng thứ tự mật độ

- Bao gồm cả các điểm mật độ thấp.
- Các điểm này chắc chắn không thuộc cụm nào.

Quá trình hình thành cụm

- Các bước kiểm tra nhiễu (noise) và thành viên cụm bị lặp lại.
- Làm tăng thời gian thực thi trong vòng lặp.

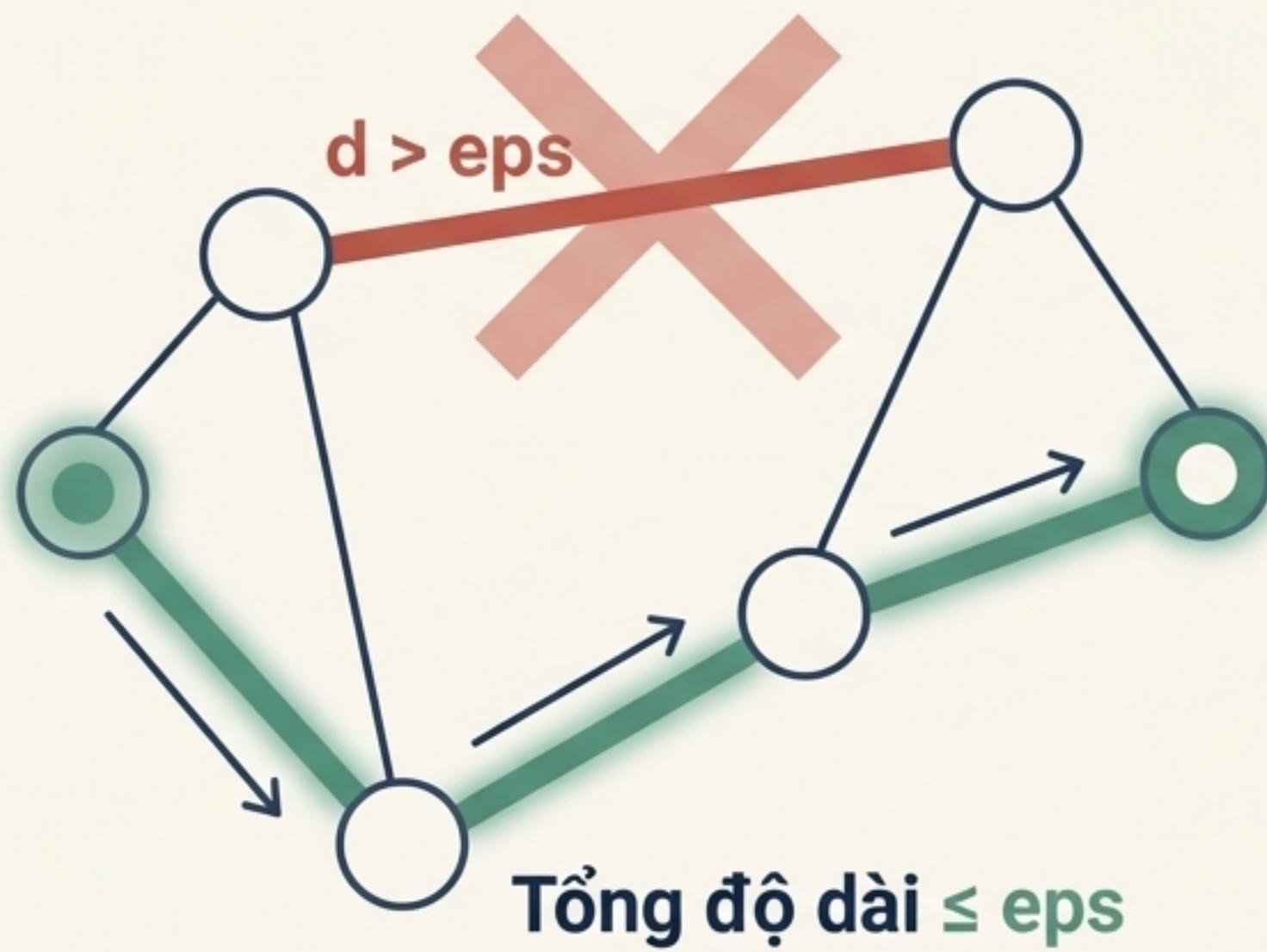
Giải pháp của chúng tôi: Một bộ công cụ tối ưu hóa thông minh



1. Cắt tỉa các cạnh không cần thiết
2. Lọc bỏ các điểm mờ độ thấp
3. Tinh giản logic thuật toán

Chúng tôi đề xuất một loạt các cải tiến nhằm thẳng vào các điểm nghẽn hiệu năng, giúp thuật toán hoạt động thông minh hơn, không cần nhiều tài nguyên hơn.

Tối ưu 1: Cắt tỉa sớm dựa trên Định lý 1



Định lý 1: Một đường đi có độ dài tối đa là eps không thể chứa một cạnh có trọng số lớn hơn eps .

Implications:

- Loại bỏ tất cả các cạnh có độ dài $> \text{eps}$ trước khi chạy thuật toán LSPD.
- Giảm đáng kể không gian tìm kiếm và số lượng tính toán.

Tối ưu 2: Lọc thông minh trong bảng mật độ



Vấn đề: Bảng thứ tự mật độ gốc chứa tất cả các điểm, kể cả những điểm có mật độ rất thấp, gây lãng phí thời gian xử lý ở bước sau.

Giải pháp:

- Chỉ đưa các điểm có mật độ $\geq \text{MinPts}$ tối thiểu vào bảng.
- Sử dụng heuristic: MinPts tối thiểu $\approx \ln(n)$, với n là tổng số điểm.

Ví dụ (HCMC Dataset): Với 10,352 điểm, chúng tôi chỉ xét các điểm có mật độ $\geq \ln(10,352) \approx 9$.

Tối ưu 3: Tinh giản Logic và Giảm thao tác thừa

Trước: Logic dư thừa

```
pseudo-code :  
point := mainhđow()  
Q.sort()  
Q.sort()  
point = Q.senetap()  
if point is noise  
    for each in point  
        Rude = point(inIsTakey())  
    end if  
end if  
for each point...  
    for each in point:  
        point = Q.cægInadsntance.mpity()  
    Q.point = pointtrain.is(noise [-2], hode)  
    return point  
end
```



Sau: Logic tinh gọn

```
pseudo-code :  
point := weighBow()  
Q.sort()  
Q.sort := Q.sort()  
point = Q.semétap()  
if point is noise  
    if each point  
        Rude = point(inIsTakey())  
    return  
end if  
for each is noise...  
    if point is point:  
        point = Q.møqušedtYixing.mpity()  
        Q.sort()  
    end if  
end
```



Key Improvements

- **Trong LSPD:** Dừng mở rộng cơ bản ngay khi đường đi đạt đến `eps`.
- **Trong tạo bảng mật độ:** Chèn điểm vào hàng đợi (queue) theo đúng thứ tự giảm dần, loại bỏ bước sắp xếp (sort) riêng biệt.
- **Trong hình thành cụm:** Loại bỏ các câu lệnh xác định nhiễu (noise) một cách thông minh. Nhiều là những điểm còn lại sau khi tất cả các cụm đã được hình thành.

Thử nghiệm trên dữ liệu thực tế: Các thành phố lớn của Việt Nam

Data Source: Open Street Map (OSM), 07/05/2020

Primary Dataset: Ho Chi Minh City:

- 10,352 Điểm ưa thích (POIs)
- 9,977 đoạn đường sau tiên xử lý

Additional Datasets: Hà Nội, Đà Nẵng, Cần Thơ, và các tỉnh khác.

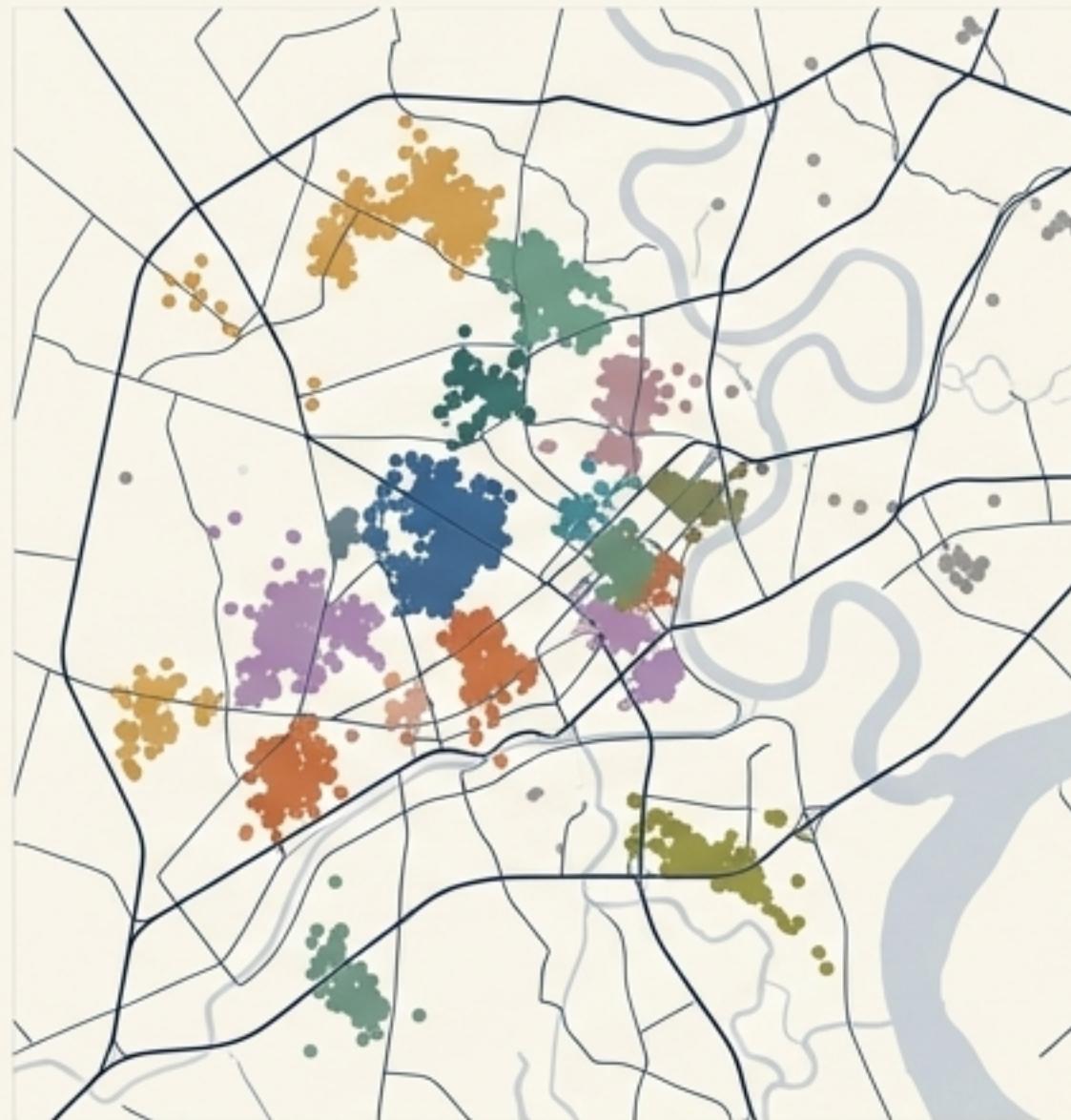


Hà Nội



Đà Nẵng

Kết quả phân cụm là hoàn toàn đồng nhất



NS-DBSCAN Gốc

Các cải tiến của chúng tôi chỉ tập trung vào hiệu năng. Kết quả đầu ra về số lượng cụm và các điểm trong cụm không thay đổi.

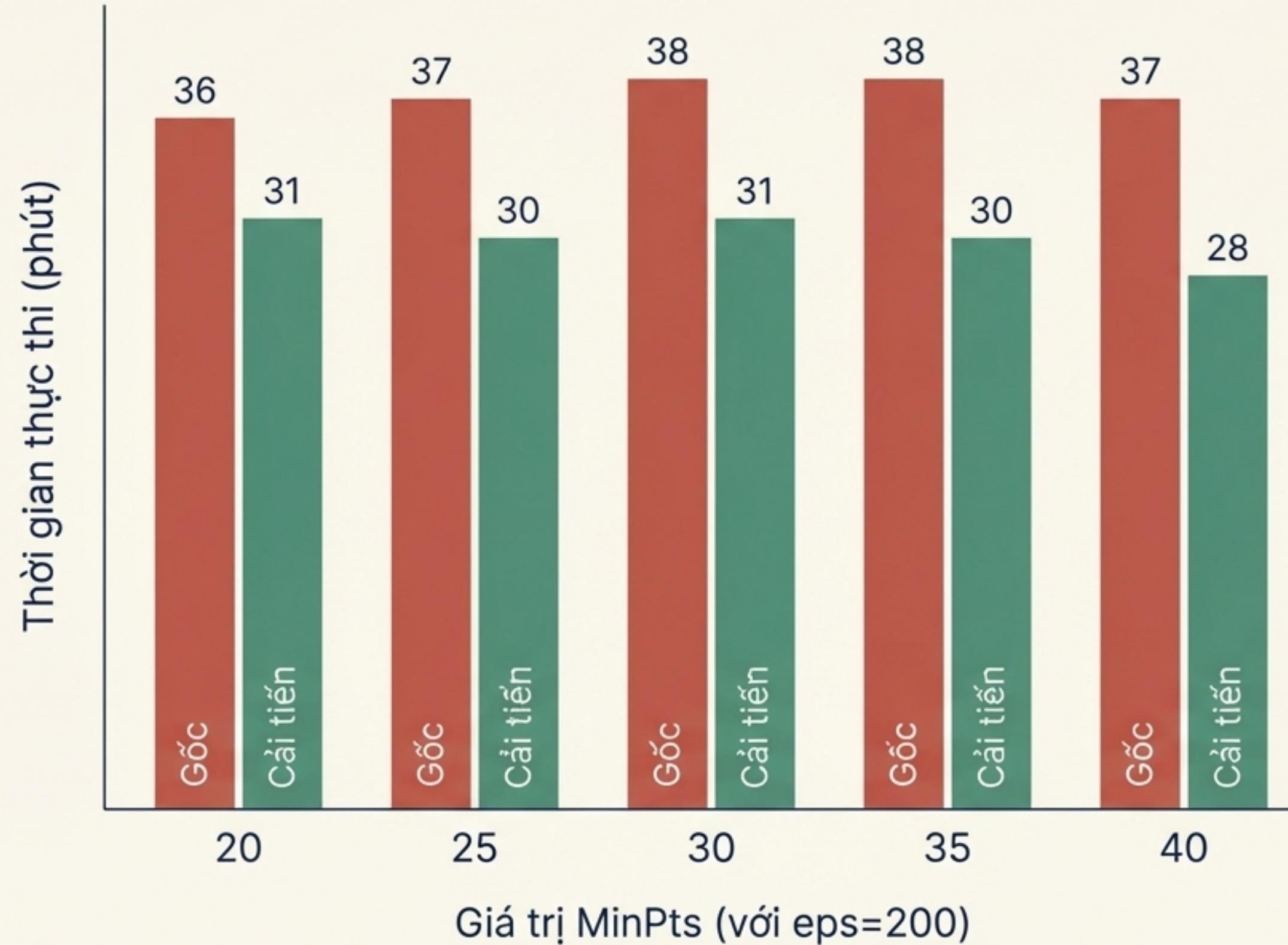
****Bảng so sánh chỉ số Silhouette cho thấy sự tương đồng gần như tuyệt đối.**

Eps	MinPts	Silhouette Score
0.5	5	0.652
1.0	10	0.621
1.5	15	0.598



NS-DBSCAN Cải tiến

Nhanh hơn từ 15% đến 25% trên dữ liệu TP.HCM



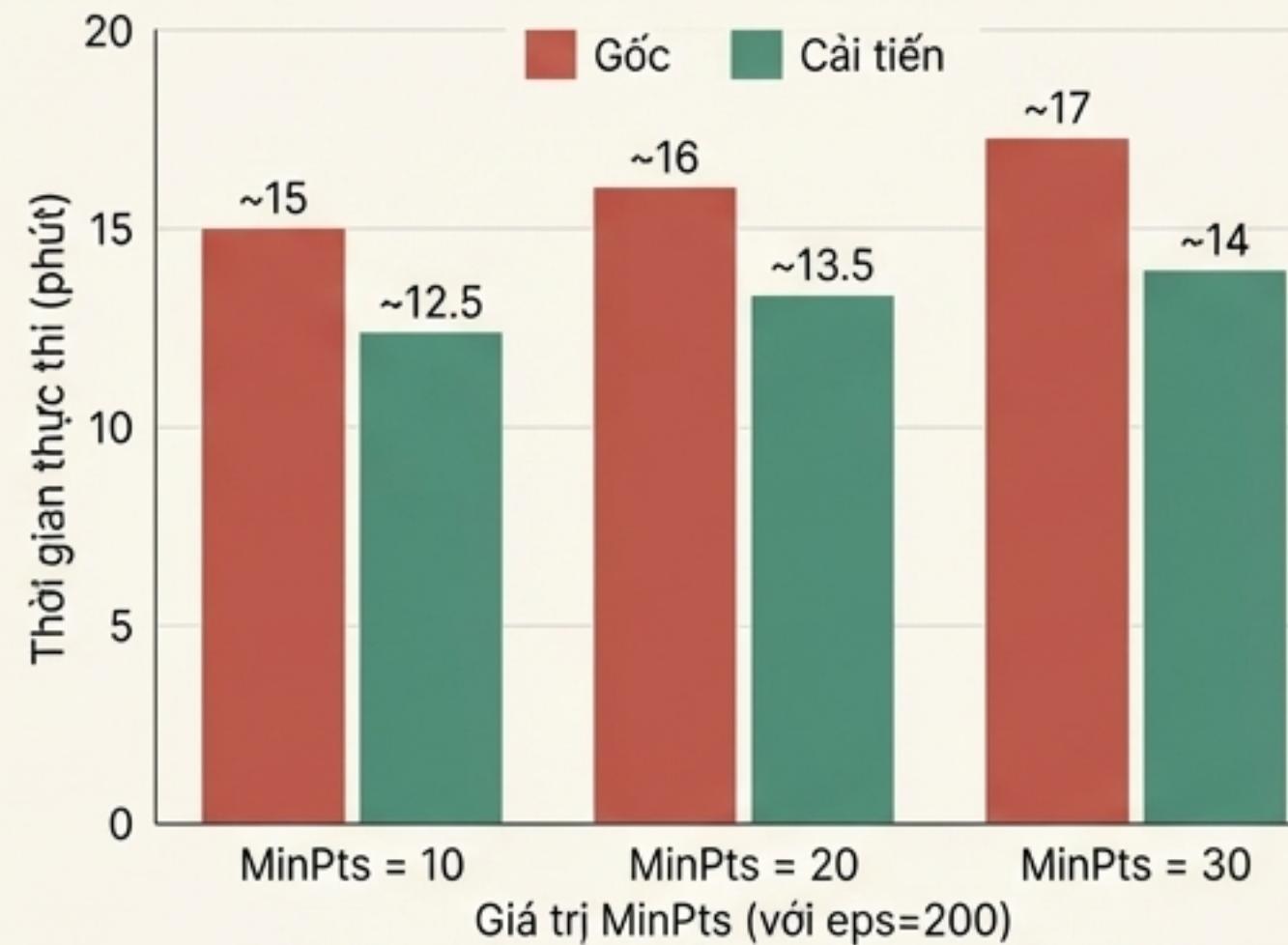
Tiết kiệm 25% thời gian

`eps=200`, `MinPts=40`

Gốc: 37 phút → Cải tiến: 28 phút

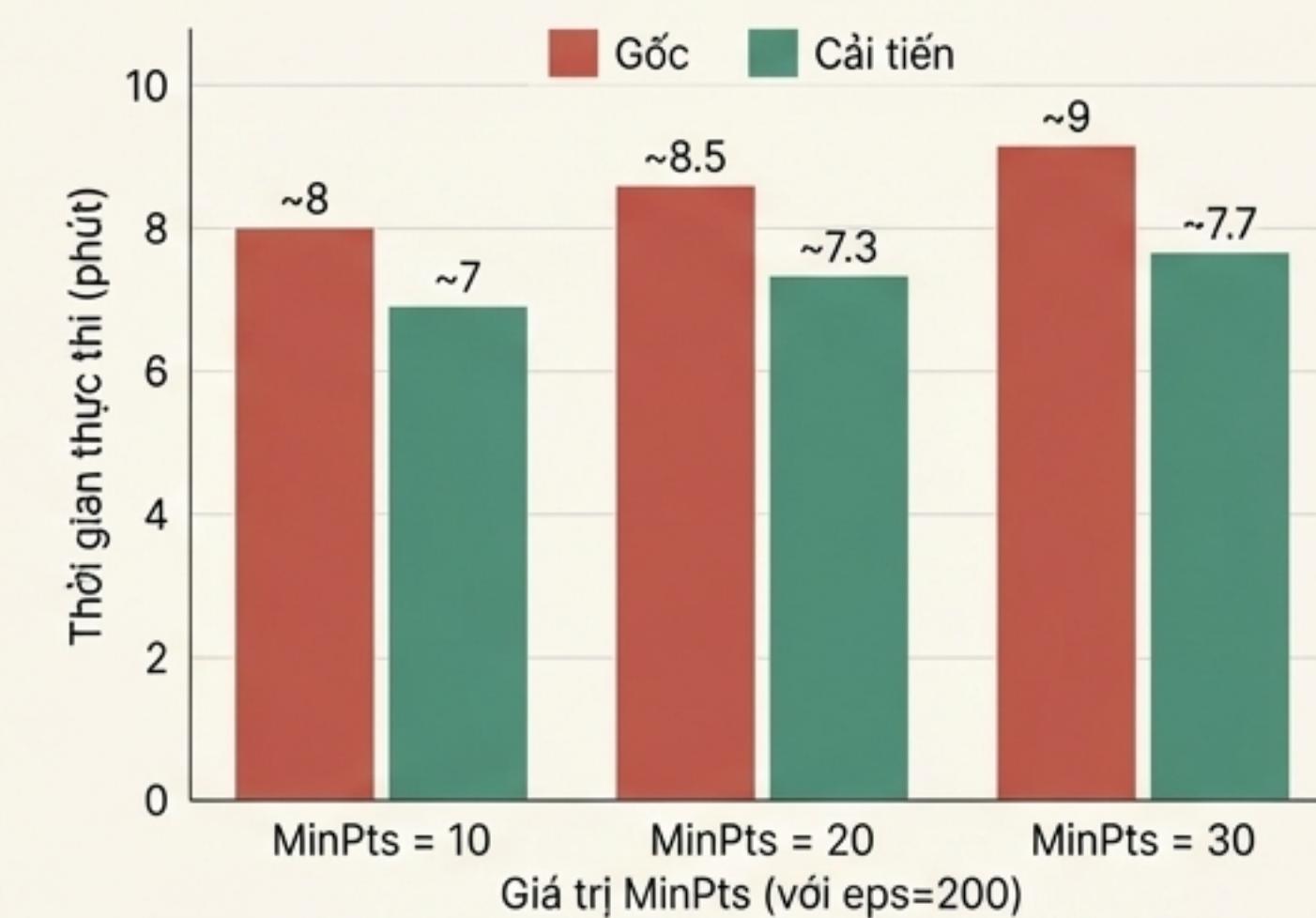
Hiệu quả nhất quán trên nhiều bộ dữ liệu khác nhau

**Hà Nội (10,273 điểm)



Giảm tới 18% thời gian thực thi.

**Đà Nẵng (3,284 điểm)



Giảm tới 15% thời gian thực thi.

Kết luận: Các cải tiến mang lại lợi ích rõ rệt trên các quy mô và đặc điểm dữ liệu đa dạng.

Đóng góp chính: Một phiên bản NS-DBSCAN hiệu quả hơn

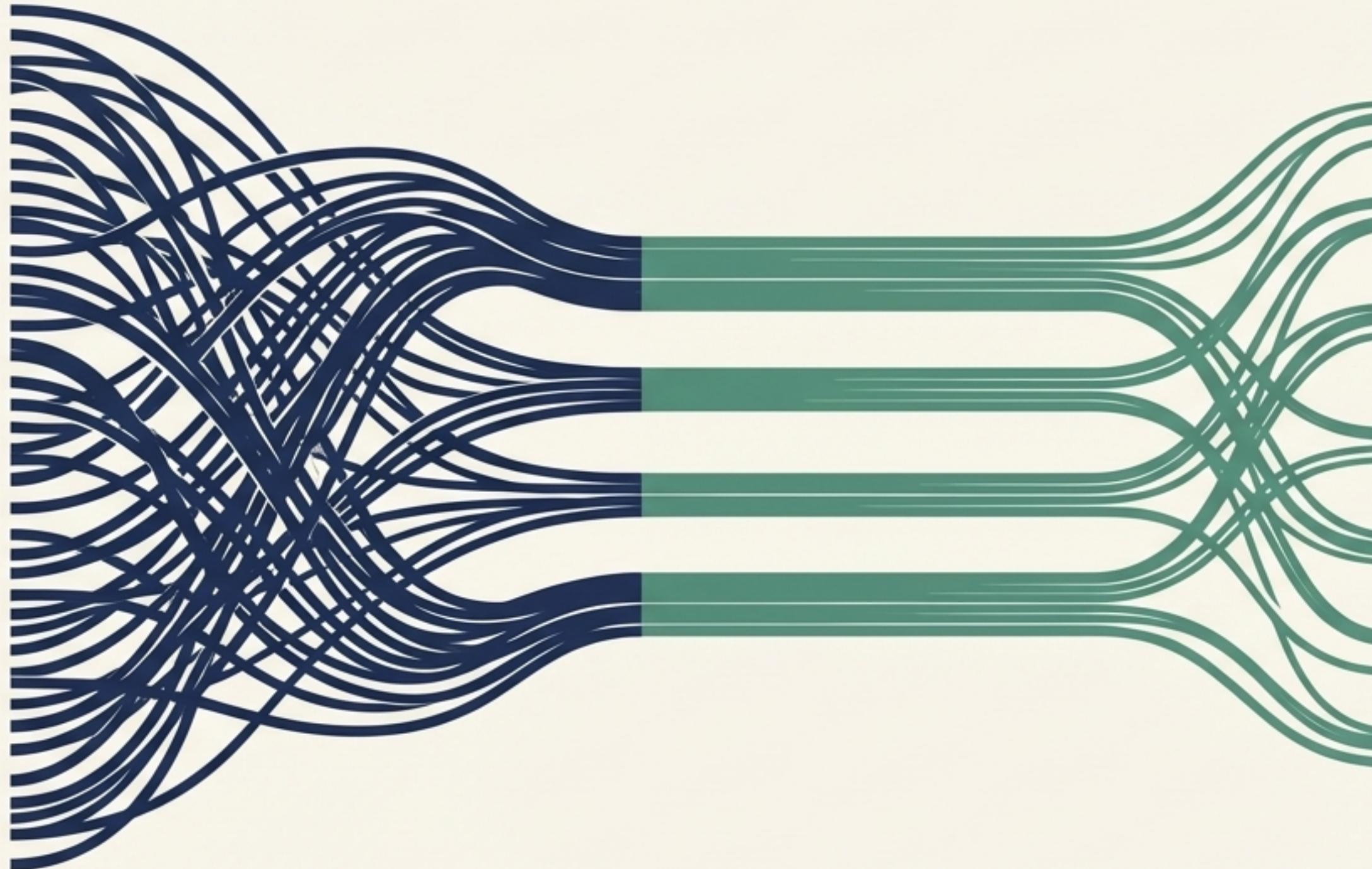


Đã xác định: Các điểm nghẽn hiệu năng cụ thể trong thuật toán NS-DBSCAN gốc.

Đã phát triển: Một bộ ba tối ưu hóa thông minh (cắt tỉa sớm, lọc mật độ, tinh giản logic) để giải quyết các vấn đề này.

Đã chứng minh: Thuật toán cải tiến nhanh hơn đáng kể (trung bình 16%) trong khi vẫn giữ nguyên hoàn toàn kết quả phân cụm.

Hướng phát triển tiếp theo: Xử lý song song cho dữ liệu cực lớn



Key Idea:

Thao tác tốn thời gian nhất là xác định các điểm lân cận.

Future Direction:

- Áp dụng kỹ thuật lập trình song song để xác định lân cận cho nhiều điểm cùng một lúc.
- Khám phá các phương pháp như MapReduce để phân tán xử lý dữ liệu.
- Mục tiêu: Giảm hơn nữa thời gian thực thi trên các tập dữ liệu quy mô toàn quốc hoặc toàn cầu.