DPS: A Novel Approach for Efficient Direction-Based Neighborhood Queries

P.H. Bergamo Bertolli [1], M. Xavier Ribeiro [1]

[1] Department of Computer Science. Federal University of São Carlos (**UFSCar**) - Brazil

Contact email: **pbertolli@estudante.ufscar.br**

IARIA

Authors

Pedro Henrique Bergamo Bertolli

He is currently a Data Engineering Specialist at Desjardins in Canada. He holds a Master's degree in Computer Science from the Federal University of São Carlos, a Specialization in Data Science from the Federal University of Technology - Paraná, and a Bachelor's degree in Computer Science from the same institution. His research focuses on the optimization of spatial queries and databases.

Dr. Marcela Xavier Ribeiro, Associate Professor

She is currently an Associate Professor at the Federal University of São Carlos (UFSCar). She holds a doctoral degree (PhD) from ICMC-USP, with a sandwich internship at Carnegie Mellon University, and a master's degree in Computer Science from UFSCar. She also earned a degree in Computer Engineering from UFSCar. She works in the Database area, addressing problems related to integration, storage, and mining of large volumes of data, images, time series, and complex data. She specializes in data mining and visualization analysis, with particular interest in spatiotemporal patterns.

Summary

01

Introduction

- Context
- Problem and Motivation

02

DPS: Our Novel Approach

- DPS goals and stages

03

Experimental Evaluation

- Datasets
- Configurations and Parametrizations
- Results

04

Conclusion

- Conclusion and future work

01

Introduction

Context

Spatial Data Utilization

- Technological advancement and popularization of mobile devices
- Large volume of spatial data generated and consumed
- Various types of applications::
  - Location-based services
  - Route planning
  - Emergency management and control

Large Data Volume

- Needs:
  - Efficient processing
  - Fast and relevant response in various contexts

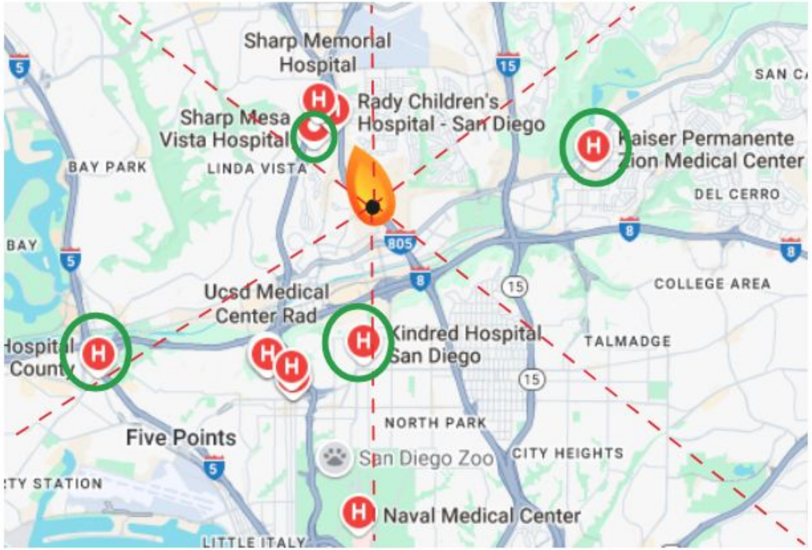
Query Contexts

- Most applications and services use the metric factor as the basis for queries
- Only distance is relevant (KNN)
- Example: Nearest hospital in relation to a person's location

Context

Query Contexts

- The metric condition alone is not always sufficient
- Directional condition is necessary, depending on the context
- Diversity in results
- Example: In a fire situation, searching for hospitals or services:
    - The closest ones might be in a direction inaccessible due to fire or smoke
    - It is desirable to have the closest services returned in different directions



Problem and Motivation

Direction and Distance-Based Neighborhood Queries

- Return the closest objects in different directions around a query point
- Examples in literature:
  - DBS (Direction-Based Surrounder)
  - DNN (Direction-Aware Nearest Neighbor)

DBS and DNN

- Based on **dominance relation**
  - Define dominant objects: those that will be returned in the response
  - Rules based on direction and distance of objects
    - Parameter θ (threshold):
      - The larger θ, the greater the restriction of results
        - Dominant interval is larger
      - The smaller θ, more comparisons performed between objects
        - Longer it takes for the stopping criterion to be reached
  - Aim to ensure **diversity of results**
- Each has its own type of dominance relation

Problem and Motivation

01 — **Computational Efficiency**

- Iterative process
- Dominance relation verified for each object until a stopping condition is reached

02 — **Result Diversity**

DBS: diverse but restricted results
DNN: more diverse results but directionally close to each other in some cases depending on database characteristics

Limitações

Need for Improvements

- Reduction in query execution time
- Consistent diversity regardless of database characteristics

03

DPS: Our Novel Approach

Direction Proximity Search (DPS)

DPS Goals

- Reduce processing time
  - Geometric partitioning of the search space that significantly accelerates nearest neighbor retrieval around query point for each partition
- Diverse and consistent results
  - Set of rules that ensures diversity and directional distribution of results regardless of database characteristics and query parameterizations

Three-stage process

Partitioning

Space surrounding the query point is partitioned into geometric regions covering different directional sectors

Processing

Identifies the nearest object to the query point within each partition

Refinement

A dominance relation is established between objects and their respective partitions, guaranteeing that each angular interval of size θ contains at most one object in the final result set

DPS - Partitioning

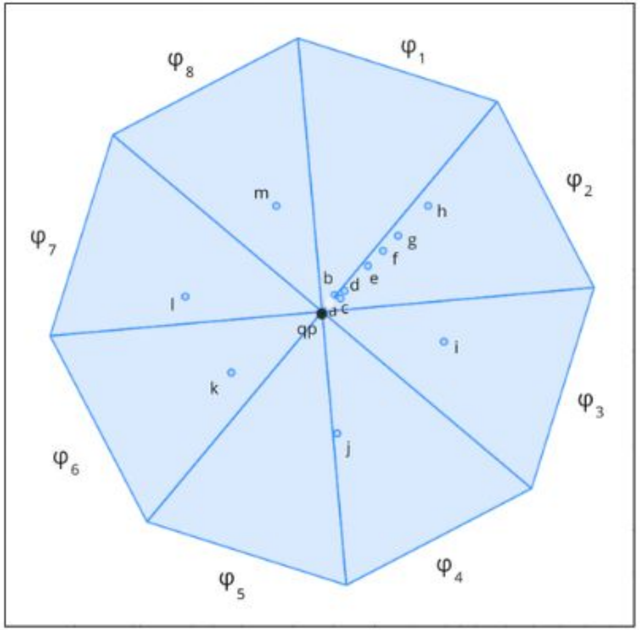
The main objective of this stage is to perform geometric partitioning of the space around the query point.

Algorithm: Partitioning

Require: Query point q, dataset D, angle θ, distance *distMax*, number of partitions n

Ensure: Set of partitions Φ

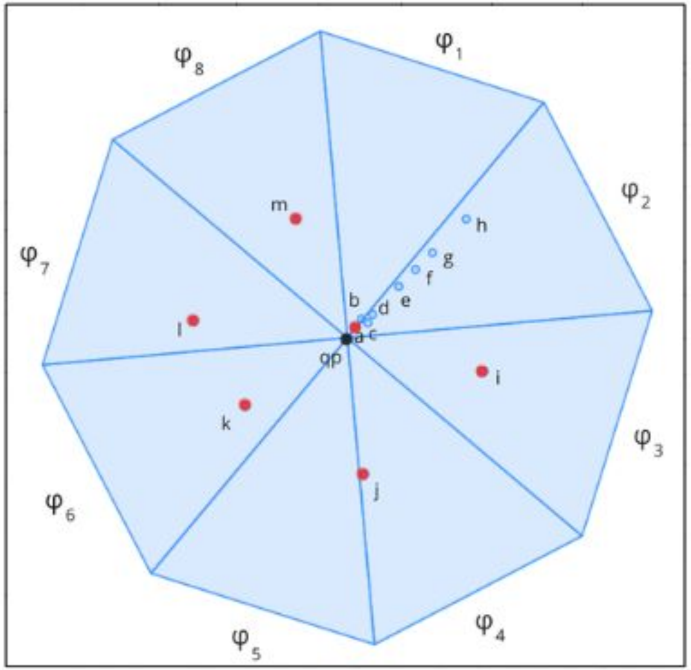
```
// First Partition Construction
NN ← FindNearestNeighbor(q, D)
NN' ← Project(NN, distMax)
v⃗ ← qNN'
ls ← Rotate(v⃗, θ/2)
φ₁ ← CreatePolygon(q, NN', ls)
Φ ← {φ₁}
// Complete Partitioning
for i = 2 to n do
  lprev ← GetUpperBoundary(φᵢ₋₁)
  lnew ← Rotate(lprev, -θ/2)
  φᵢ ← CreatePartition(lprev, lnew)
  Φ ← Φ ∪ {φᵢ}
end for
return Φ
```



DPS - Processing

The objective of this stage is to process each partition simultaneously and return the objects closest to **q**

- Retrieves objects that intersect with the partition
- Returns the object closest to q in each partition



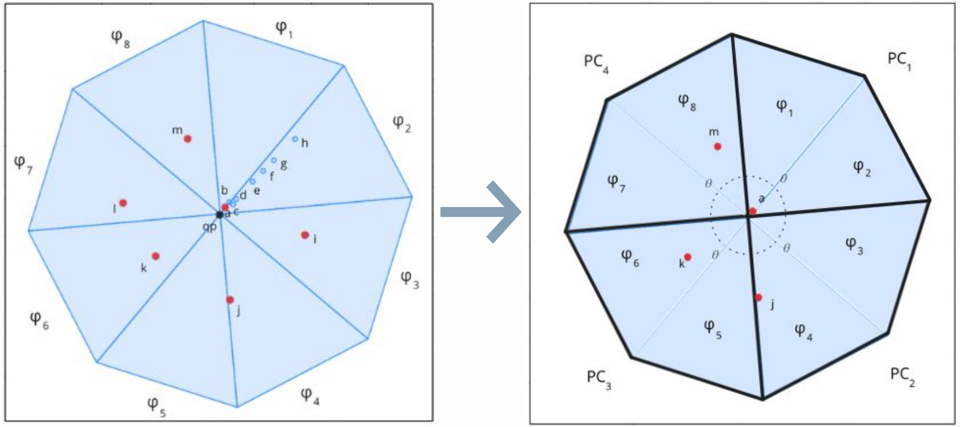
DPS - Refinement

- The objective of this stage is to ensure that the objects found are not directionally close to each other.
- Reduce the number of partitions to guarantee that each θ interval is represented by at most 1 object.

Algorithm: Refinement

Require: Ordered processing list $List_p$
Ensure: Result set R

```
ignored ← ∅
R ← ∅
for each (φᵢ, NNᵢ, distᵢ) in Listₚ do
if φᵢ ∉ ignored then
  R ← R ∪ {NNᵢ}
  adjacent ← GetAdjacentPartitions(φᵢ)
  ignored ← ignored ∪ adjacent
end if
end for
return R
```



04

Experimental evaluation

Datasets

Synthetic Datasets

- Spider Generator
- Volumes
  - Small: 20.000
  - Medium: 200.000
  - Large: 2.000.000
- Distributions
  - Uniform
  - Diagonal
  - Gaussian
  - Sierpinski
  - Bit

Real Datasets

- Collected from the OpenStreetMap platform
- Extraction of point-type data from Brazil
- Volumes
  - Small: 22.628
  - Medium: 174.449
  - Large: 3.733.292

Configurations and Parameterizations

Hardware Configurations

- Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 12 cores
- 16GB RAM
- SSD 1TB
- Ubuntu 20.04.1 64 bits

Query Parameterization

- Variation of θ (20°, 45°, 60° and 90°)
- Variation of **query point** and **distMax** between synthetic and real databases
- Cache cleaning after each query

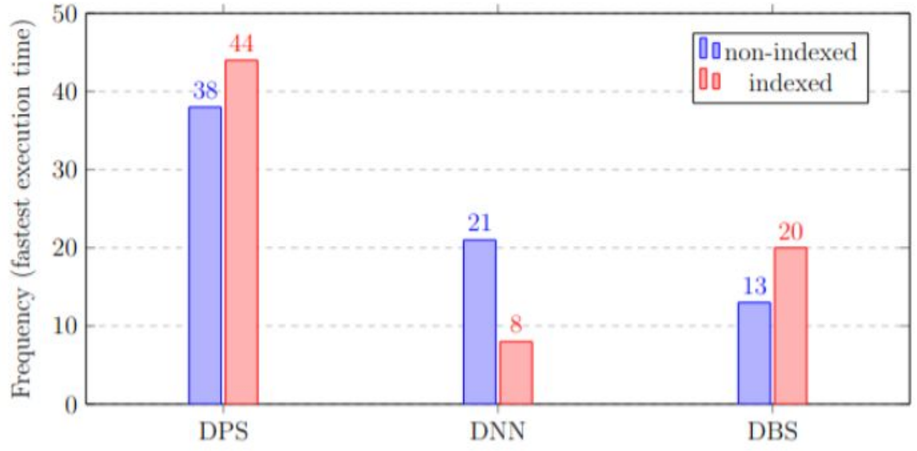
Evaluation process

- Execution of queries on non-indexed databases
- Execution of queries on databases indexed using the PostGIS GiST method (R-Tree variant)
- Results analyzed to identify the best execution times for each type of parameterization
- Comparative analysis of the diversity of objects returned by queries

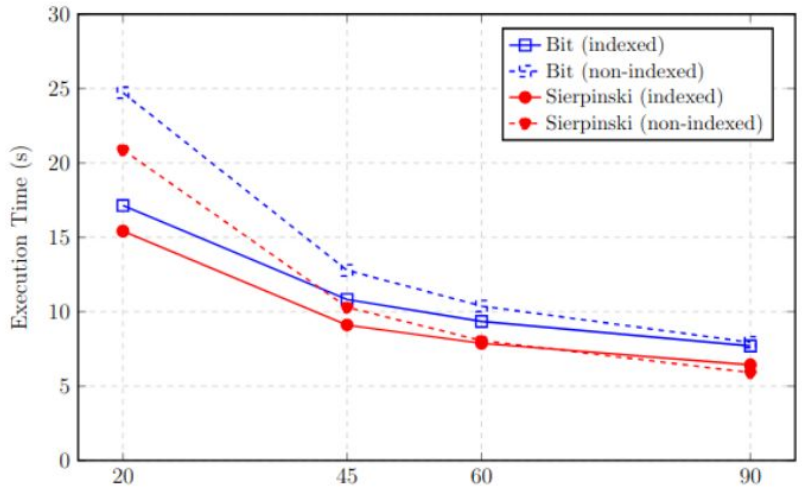
Results

Performance Overview

- Best execution times
  - 52.8% of queries on non-indexed databases
  - 61.1% of queries on indexed databases

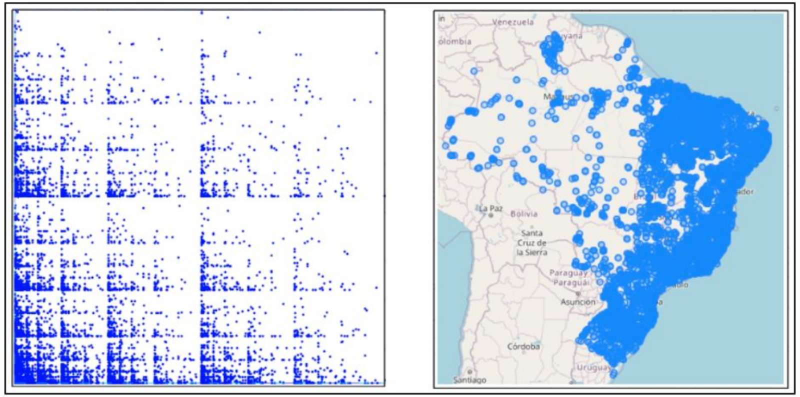


- Execution time for large datasets of Bit and Sierpinski distributions
  - Up to 99.9% improvement in execution time



Results

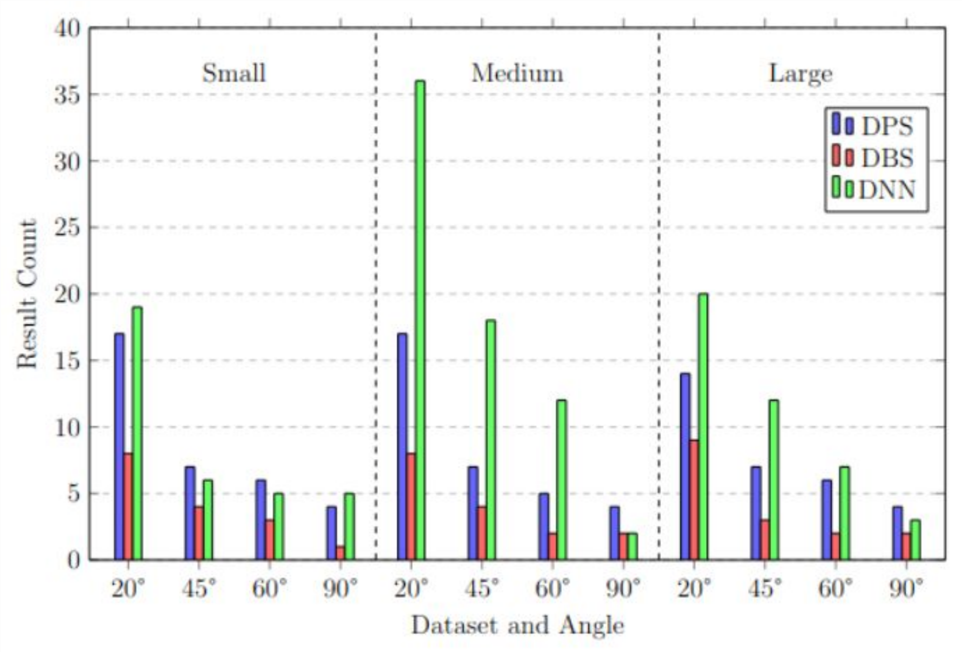
- Real Datasets:
  - Real data has similar pattern as Bit distribution (high direction density)
  - Only 2 cases where it didn't have the best result
  - Best result: Average improvement of 97.3% in execution time for 20-degree angle in medium-sized database



Database Size	Angle	Non-Indexed			Indexed		
		DBS	DNN	DPS	DBS	DNN	DPS
Small	20°	4.37	4.36	**3.14**	4.19	4.22	**1.26**
	45°	2.18	2.13	**1.59**	2.00	2.05	**0.80**
	60°	1.25	**1.24**	1.34	1.13	1.14	**0.78**
	90°	1.28	1.18	**0.87**	1.11	1.19	**0.66**
Medium	20°	187.87	183.04	**4.51**	177.45	182.37	**5.05**
	45°	21.84	21.81	**2.39**	21.30	22.16	**2.51**
	60°	6.04	6.05	**1.92**	5.92	6.01	**2.09**
	90°	**0.78**	**0.76**	1.50	**0.71**	**0.71**	1.63
Large	20°	93.44	98.22	**37.41**	91.63	98.58	**32.13**
	45°	74.59	73.90	**19.06**	72.57	73.02	**19.44**
	60°	33.97	33.85	**15.27**	32.80	32.50	**16.75**
	90°	4.22	**3.97**	11.61	3.52	**3.49**	13.49

Results

- Diversity and consistency of the result set
- One dominant object per θ interval
- Maximum of 360°/θ results:
    - 20° = 18
    - 45° = 8
    - 60° = 6
    - 90° = 4



Results

- **Overall Results:**
  - DPS outperformed DBS and DNN algorithms in 52.8% of queries on non-indexed databases and 61.1% on indexed databases.
  - With Bit and Sierpinski distributions, DPS achieved exceptional performance with 99.9% improvement in execution time on large databases.
  - The algorithm excels in spatial distributions with high object density in specific directions, explaining its superior performance on both synthetic and real-world datasets.
  - DPS showed better performance with smaller angle parameters (20° and 45°) as they impose less strict dominance restrictions.
  - The algorithm consistently maintains that the maximum number of returned objects equals 360°/θ, guaranteeing at most one dominant object per θ interval.

05

Conclusion

Conclusion

- We presented DPS (Direction Proximity Search), a geometric partitioning approach for direction-based queries. DPS reduces execution time by up to 99.9% compared to existing methods—completing queries in under 25 seconds that previously took over 24 hours.
- It ensures directional diversity by returning at most one object per θ interval.
- While our experiments focused on geographic datasets, DPS has potential applications in autonomous navigation, IoT networks, and emergency response scenarios.
- Our evaluation was limited to datasets of up to 2 million points, with real-world applications potentially presenting additional challenges.

Future work:

- Intelligent Query Selection: Develop models to automatically choose between DPS, DBS, or DNN based on dataset characteristics and query parameters.
- Scalability Analysis: Evaluate DPS performance with datasets exceeding 10 million objects and identify optimization opportunities.
- Dynamic Environments: Adapt DPS for scenarios with frequently changing data, such as real-time traffic or mobile sensor networks.
- Extended Domains: Explore applications beyond spatial queries, including similarity searches in high-dimensional spaces.