DISCOVERING PREVALENT CO-LOCATION PATTERN IN DIFFERENT DENSITY SPATIAL DATA WITHOUT DISTANCE THRESHOLDS

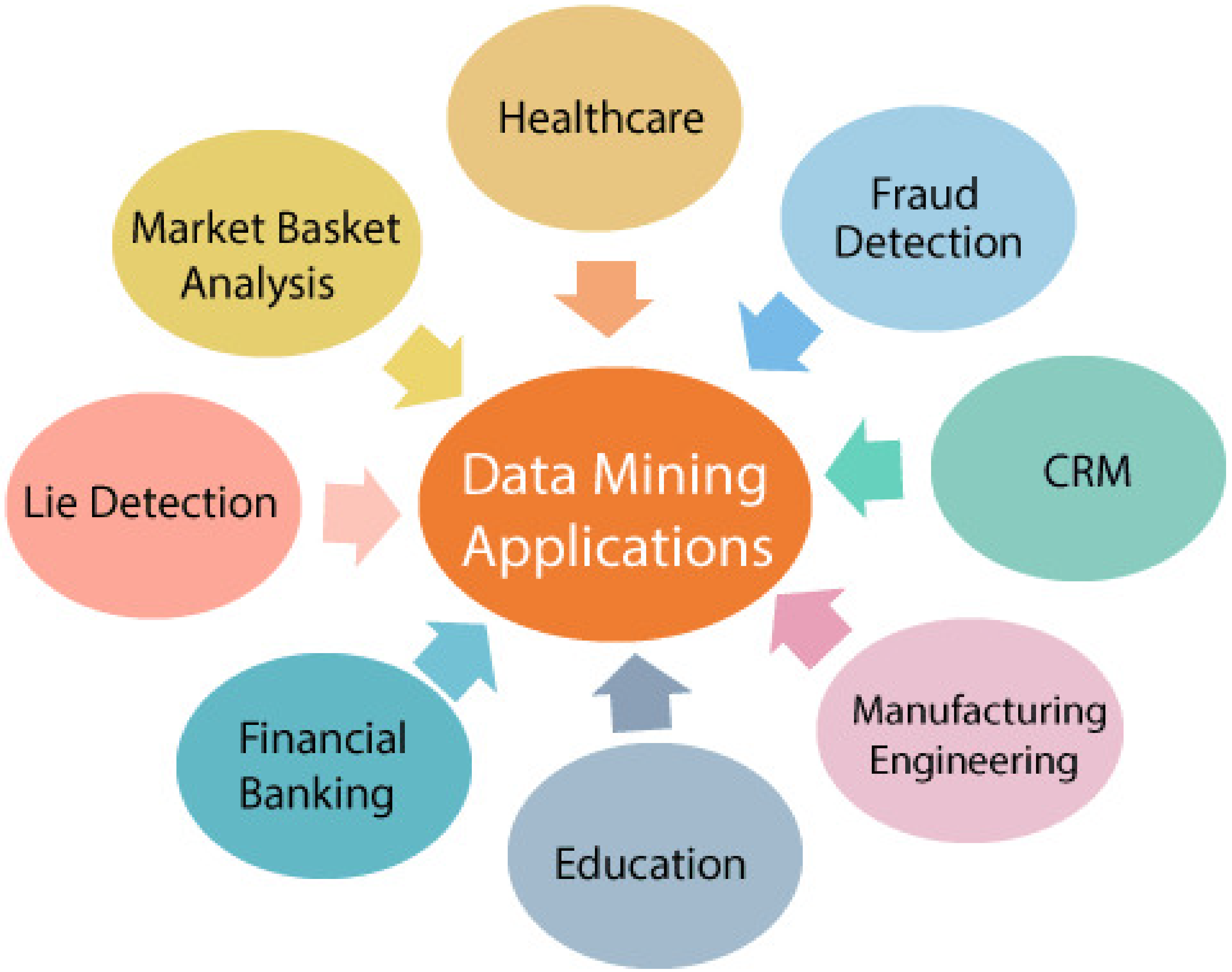
Tran Duy Hai
Le Anh Thang
Ha Trong Nguyen

Advisor: M.S.E Le Dinh Huynh

Table of content

Introduction

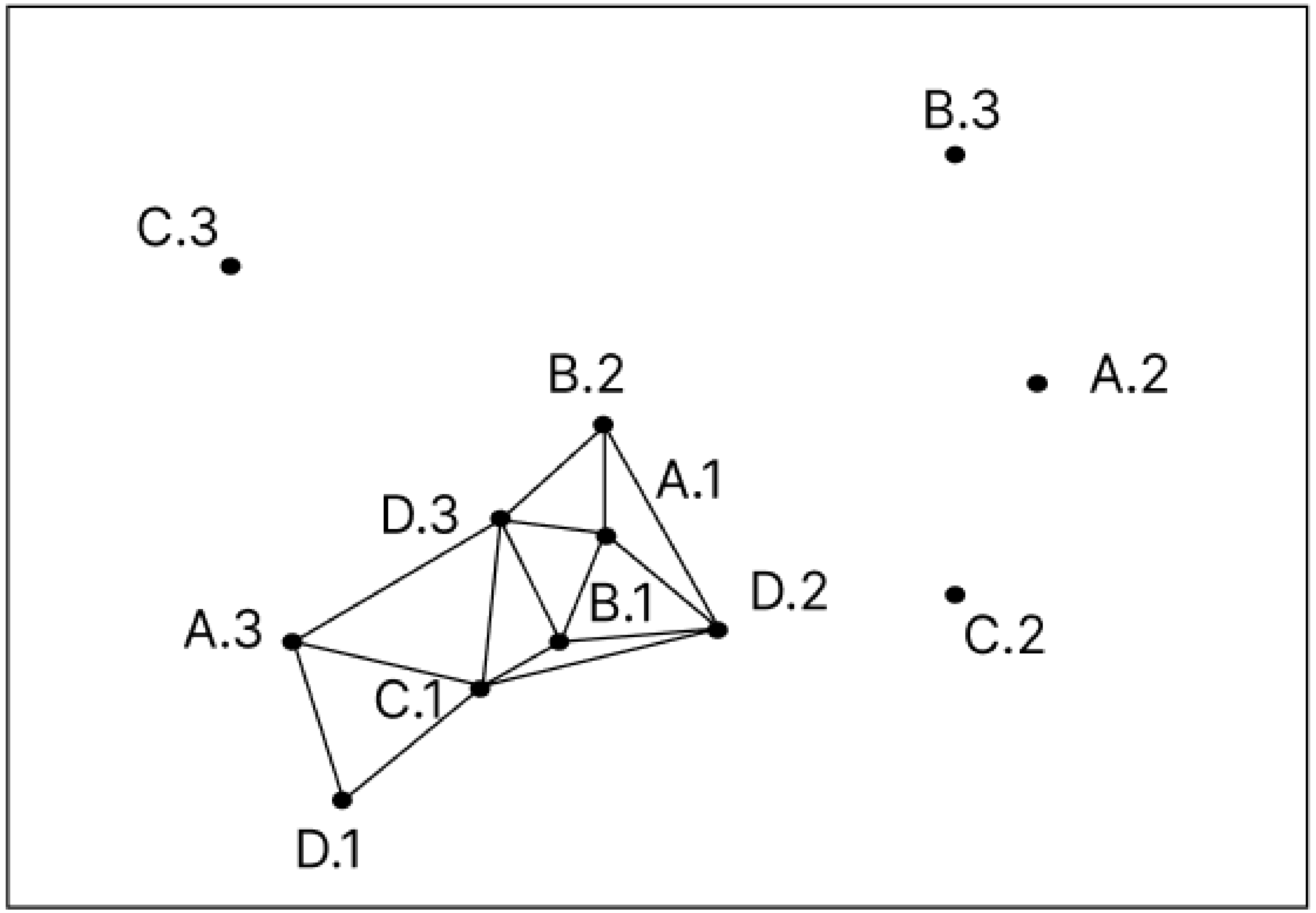
Prevalent spatial co-location patterns (PSCPs) is aiding in forecasting and making informed decisions while optimizing resource management and allocation.



Introduction

Canculate Prevalent spatial co-location pattern:

Given a spatial dataset S = {S1,...,Sn} where Si = {fi1 ,...,fim }, (1 ≤ i ≤ n) is a set of instances of a feature fi



Canculate Prevalent spatial co-location pattern:

- Participation ratio, PR

$$PR(c, f_i) = \frac{\text{Number of distinct instances of } f_i \text{ in } c}{\text{Number of instances of } f_i \text{ in } S}$$

Canculate Prevalent spatial co-location pattern:

- Participation index, PI

$$PI(c) = \min_{f_i \in c}\{PR(c, f_i)\}$$

Canculate Prevalent spatial co-location pattern:

- Prevalent spatial co-location pattern

$$PI(c) \geq min_{prev}$$

Research Methods

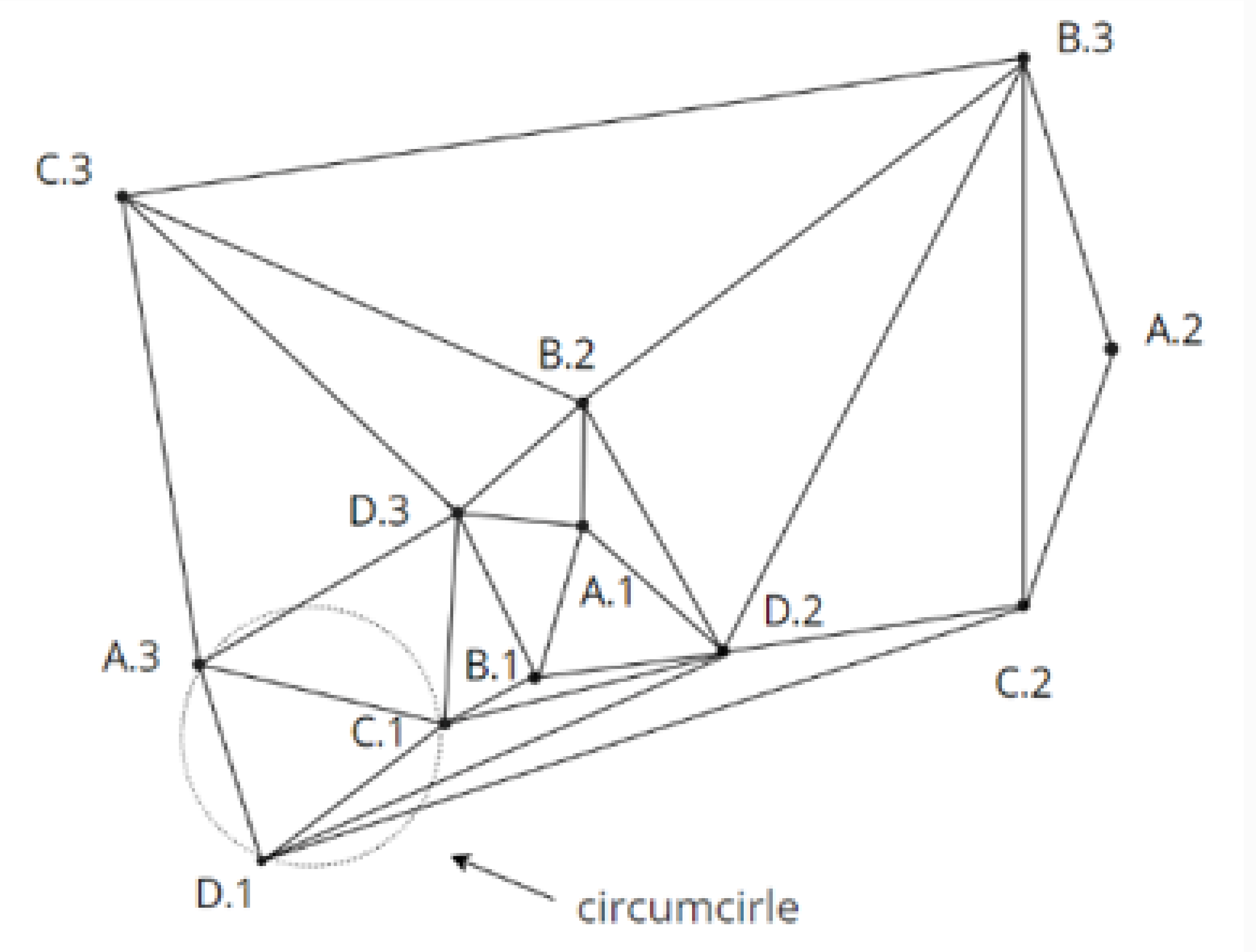
Methods use distance threshold	Disadvantage
Joinless approach	Still needs to handle a large number of candidate co-location instances and compute related statistics
Patial join approach	Also requires handling the conversion of data into transaction format and computing case joins.
Co-location pattern instance-tree	Still face scalability challenges when dealing with large and dense datasets
Improve Co-location pattern instance-tree	The computational complexity can increase significantly as the dataset size grows
Mapreduce	Distance threshold is used as a parameter for the process of searching co-location patterns.

Previously proposed method

Methods not use distance threshold:

※ Delaunay method

This method can arise when the connecting edges in the Delaunay triangulation are excessively long or when they connect instances with identical feature types.



Previously proposed method

Methods not use distance threshold:

※ DT-based co-location pattern mining method (DTC)

It is creating candidate colocations using the Association Rule method in the DTC algorithm results in a large number of candidates

➔Excessive memory

Previously proposed method

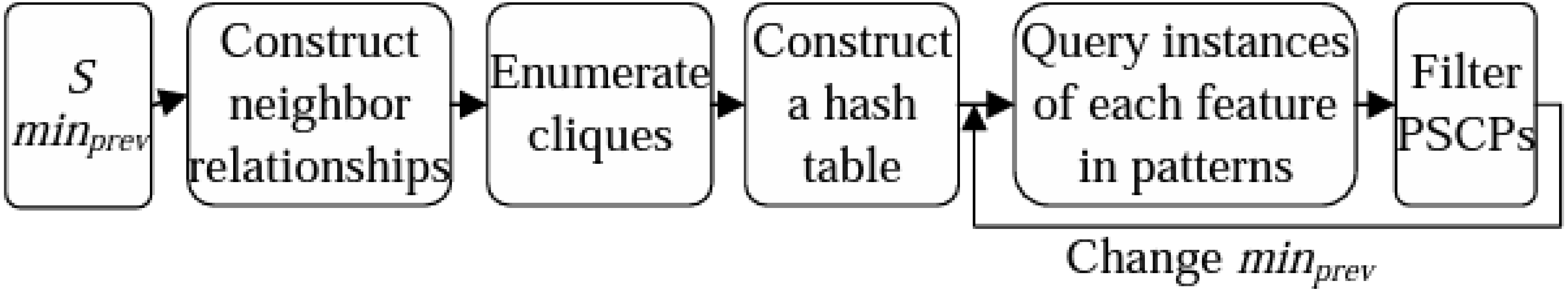
Methods not use distance threshold:

※ A clique-based approach for co-location pattern mining

- IDS is suitable for processing large but sparse data
- NDS is designed for dense but not large data

→ When dealing with large and dense datasets, IDS requires long processing time, while NDS causes memory overflow

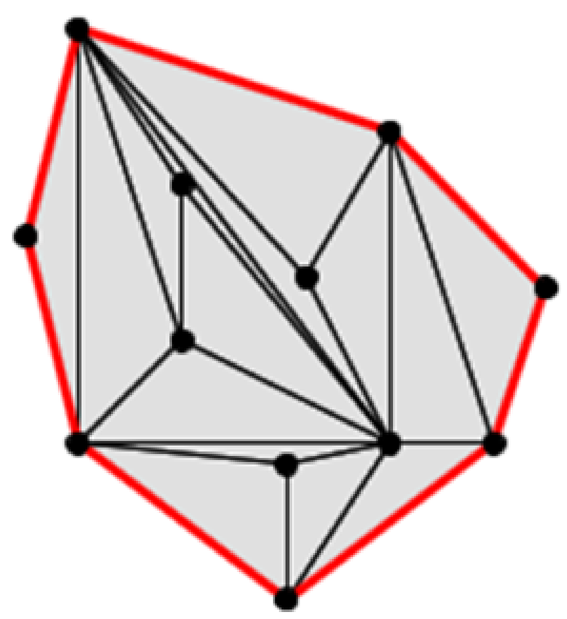
Research Methods



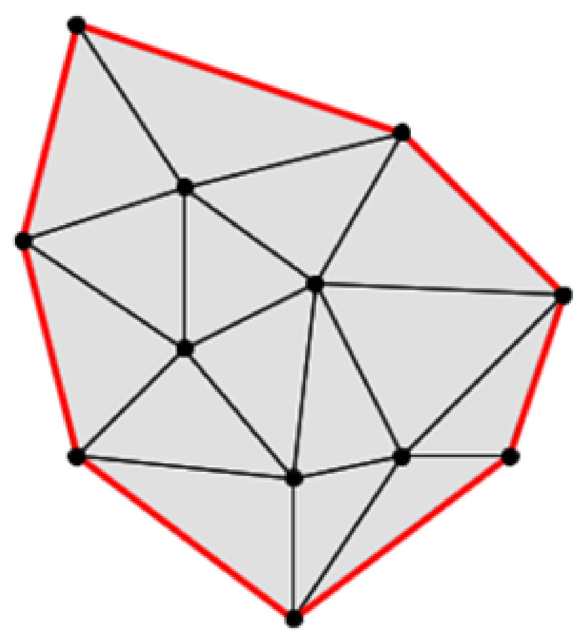
DTk-orderclique-basedminingPSCPalgorithm.

Delaunay Triangle

- DT is a geometric structure that consists of a set of non-overlapping triangles.
- DT is constructed in suck way that there is no point lies within the circumcircle of any triangle in the triangulation



non-DT construct



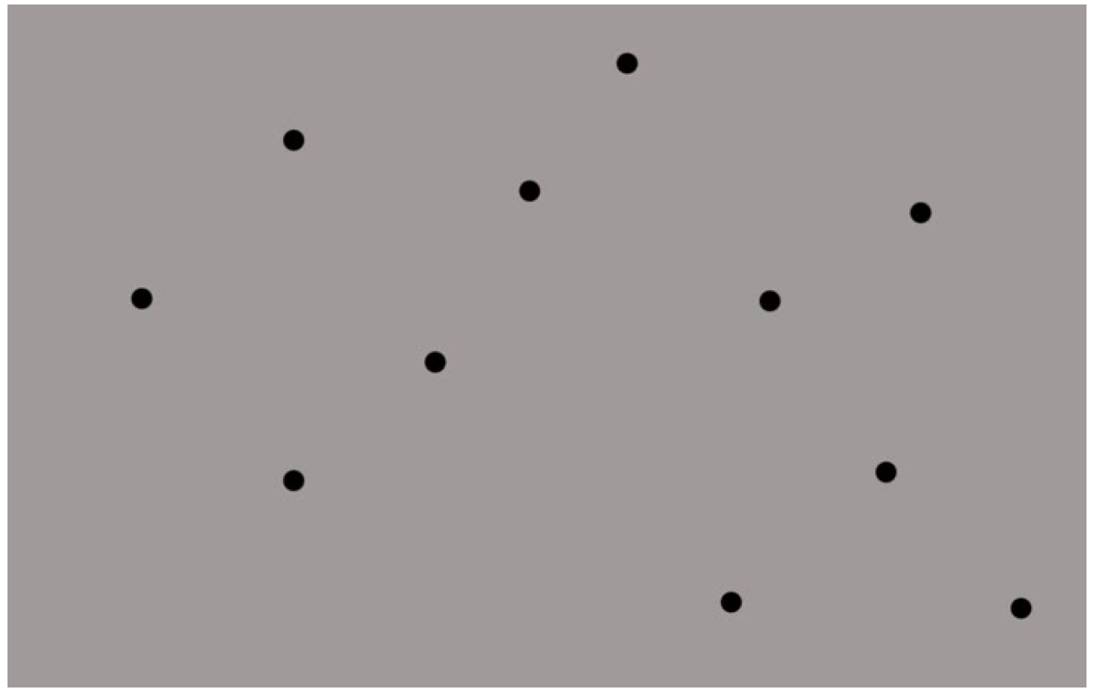
DT construct

Delaunay Triangle Algorithms

- Algorithms with O(nlogn) complexity

Sweep Line Algorithm

Incremental Algorithm

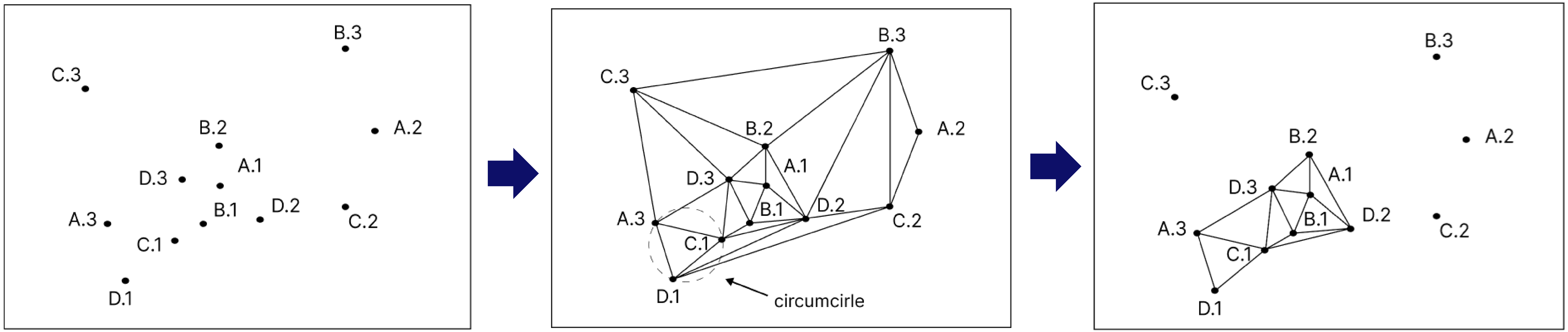


Divide and Conquer Algorithm

Delaunay Triangle

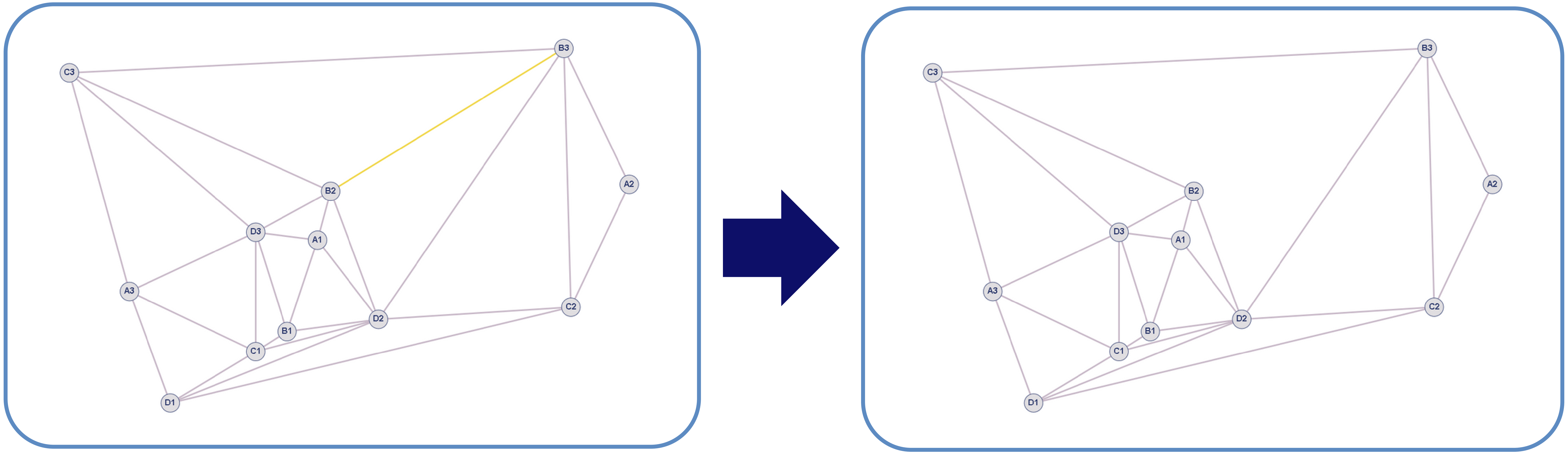
After using DT algorithm, there are edges that should not be connected. Then there are 3 filters to remove their edges.

- Filter to remove Feature edges.
- Filter to remove Global edges.
- Filter to remove Local edges.



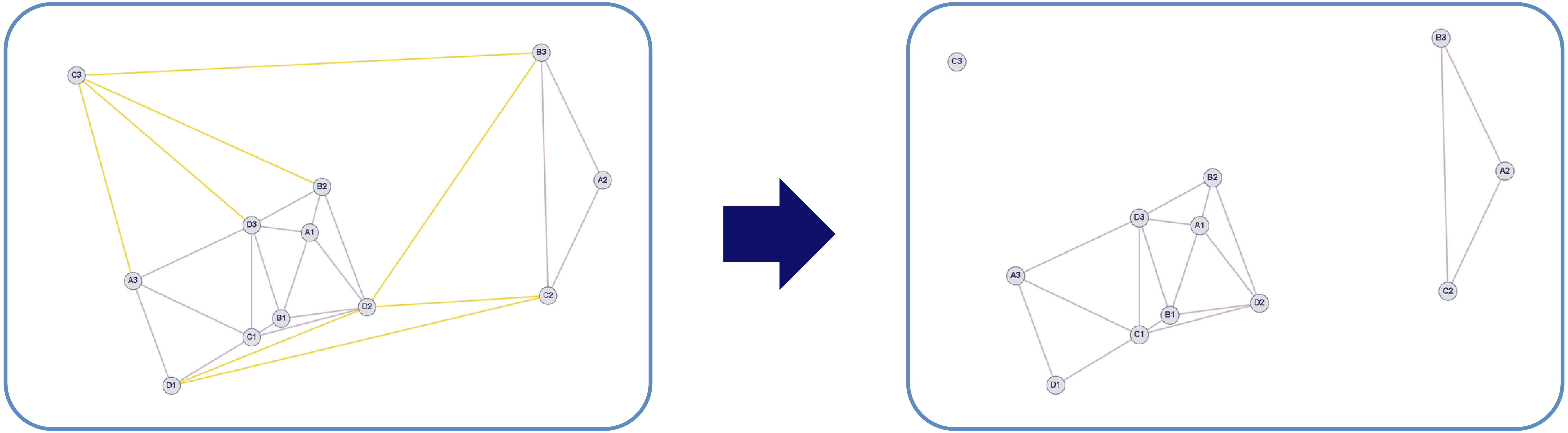
Delaunay Triangle Constraint

1. Feature constraint



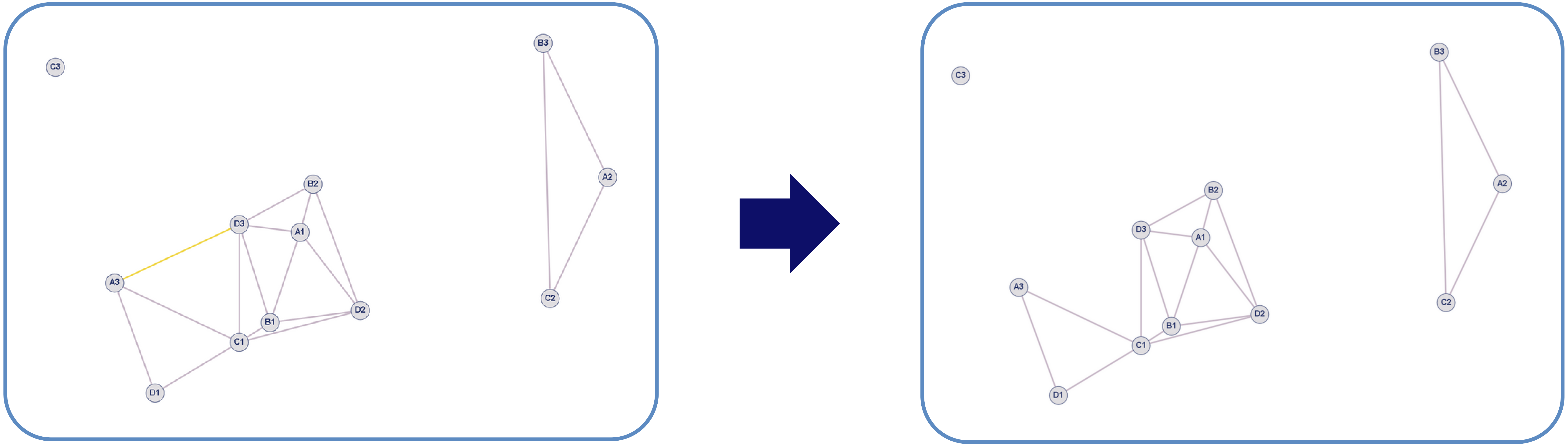
Delaunay Triangle Constraint

2. Global edge constraint



Delaunay Triangle Constraint

3. Local edge constraint

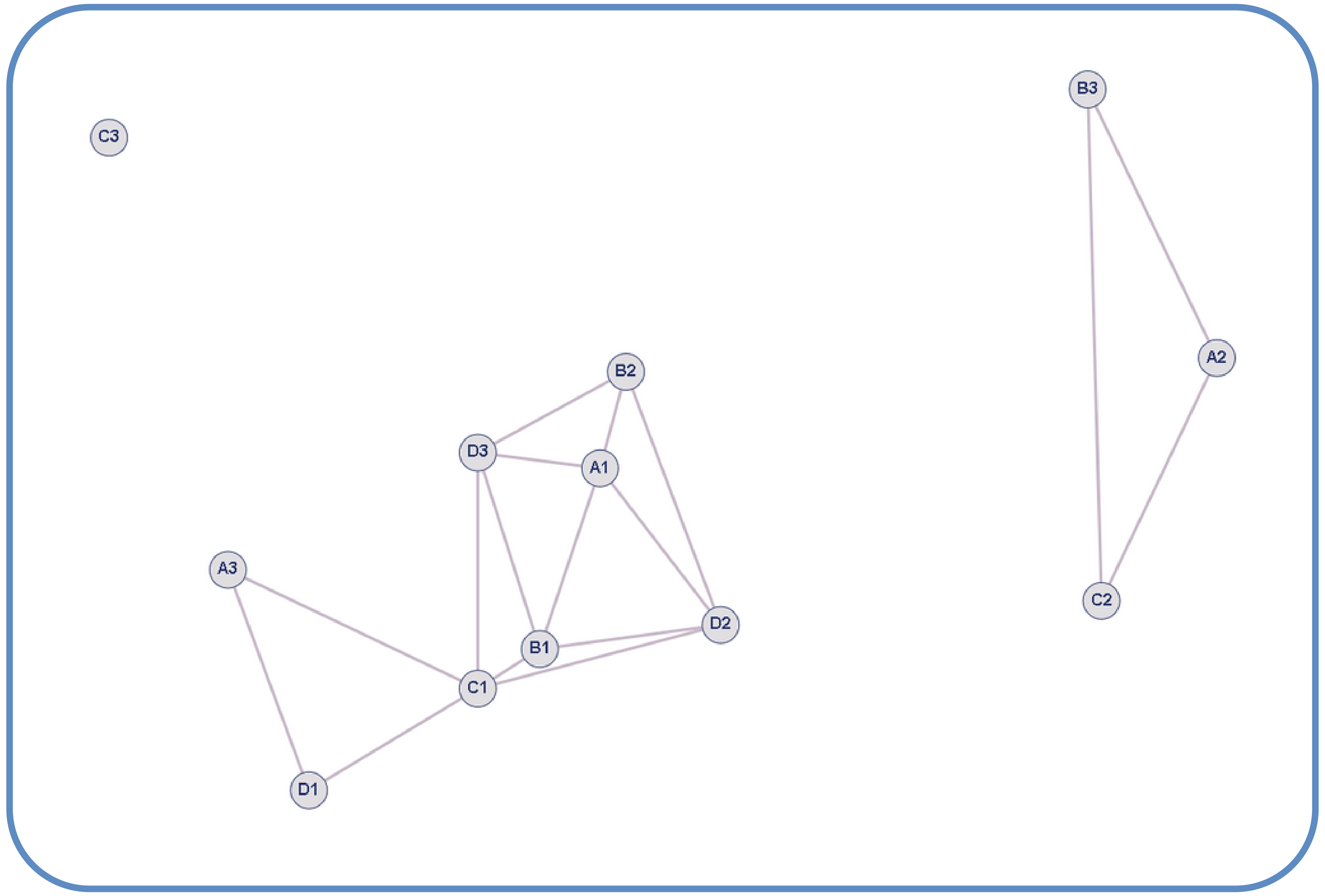


K-order neighbors approach

After a DT-based approach, we gain data about the edges of dataset S. With each edge, we can conduct first-order neighbors of each point in S.

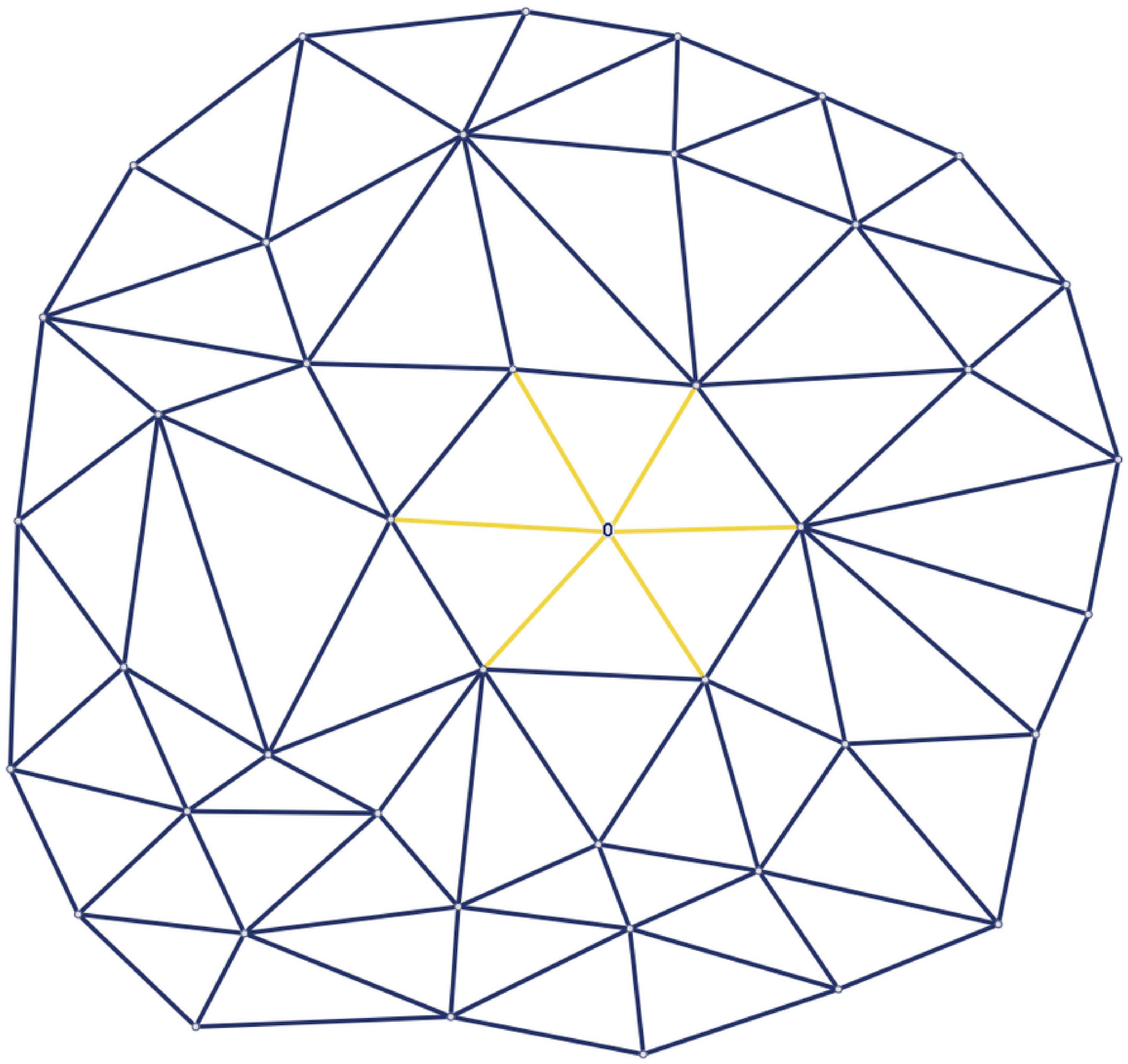
Example:

- **Edges:** A3C1, D1C1, A3D1, C1B1, D3A1, B1A1, ...
- **First-order neighbors:**
  - A3: D1, C1
  - D1: A3, C1
  - C1: A3, D1, B1, D3, D2



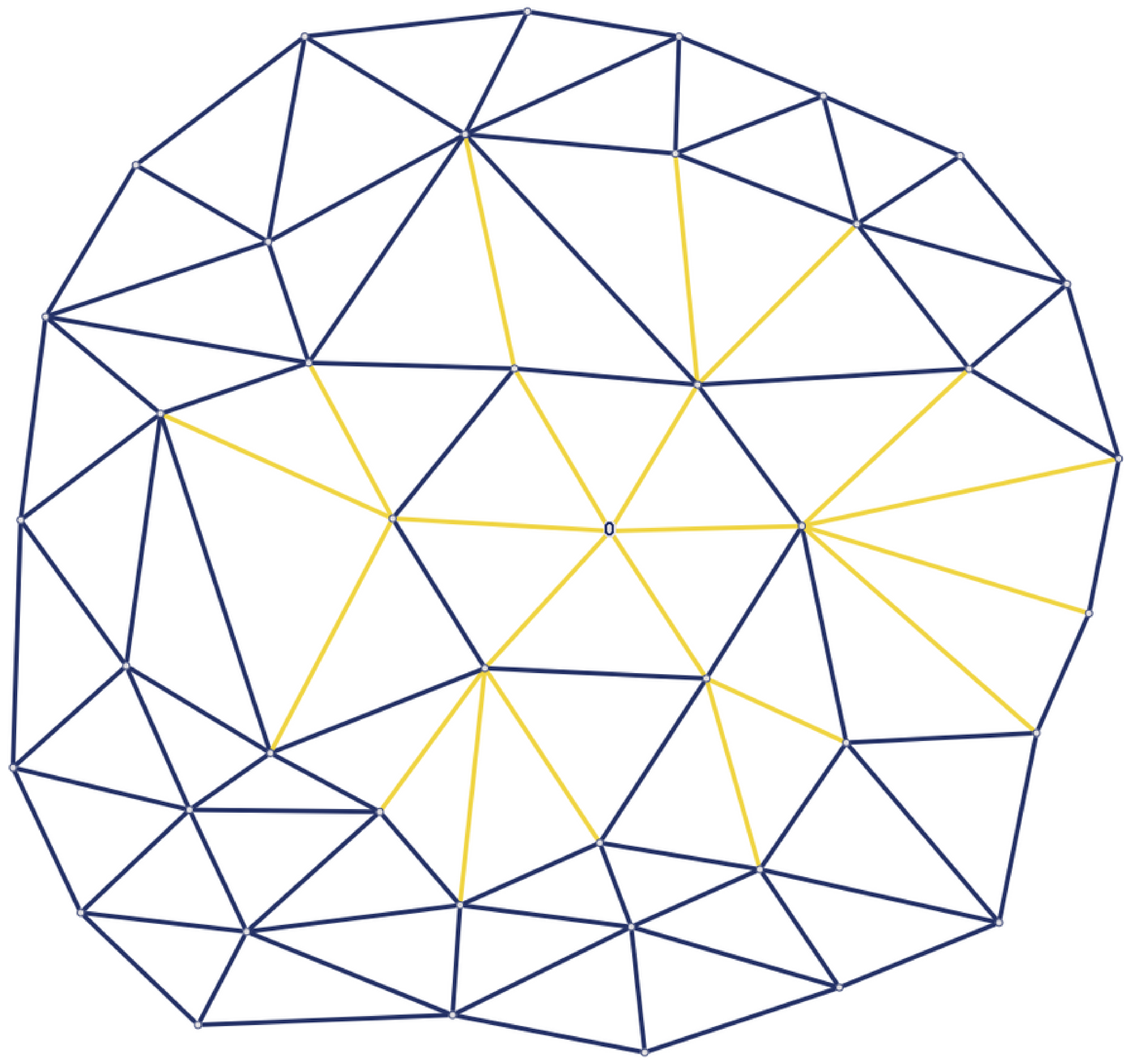
K-order neighbors approach

K-order neighbors: Those points directly connected to a point v by the edges of the DT are called first-order neighbors' of this point.Then, those points which are directly connected to the first-order neighbours and are not first-order neighbours themselves, are called second-order neighbors. Continuing this process, we get k-order neighbours for point v.



K-order neighbors approach

K-order neighbors: Those points directly connected to a point v by the edges of the DT are called first-order neighbors' of this point.Then, those points which are directly connected to the first-order neighbours and are not first-order neighbours themselves, are called second-order neighbors. Continuing this process, we get k-order neighbours for point v.



K-order neighbors approach

Big neighborhood (BO): Big neighborhood of instance s is the instances s' that satisfy both the relationship R(s', s) with s and belong to sets with higher indices of feature than the set containing instance s.

Instance	BNs	Instance	BNs	Instance	BNs	Instance	BNs
A.1	B.1, B.2, D.2, D.3	B.1	C.1, D.2, D.3	C.1	D.1, D.2, D.3	D.1	-
A.2	B.3, C.2	B.2	D.2, D.3	C.2	-	D.2	-
A.3	C.1, D.1	B.3	C.2	C.3	-	D.3	-

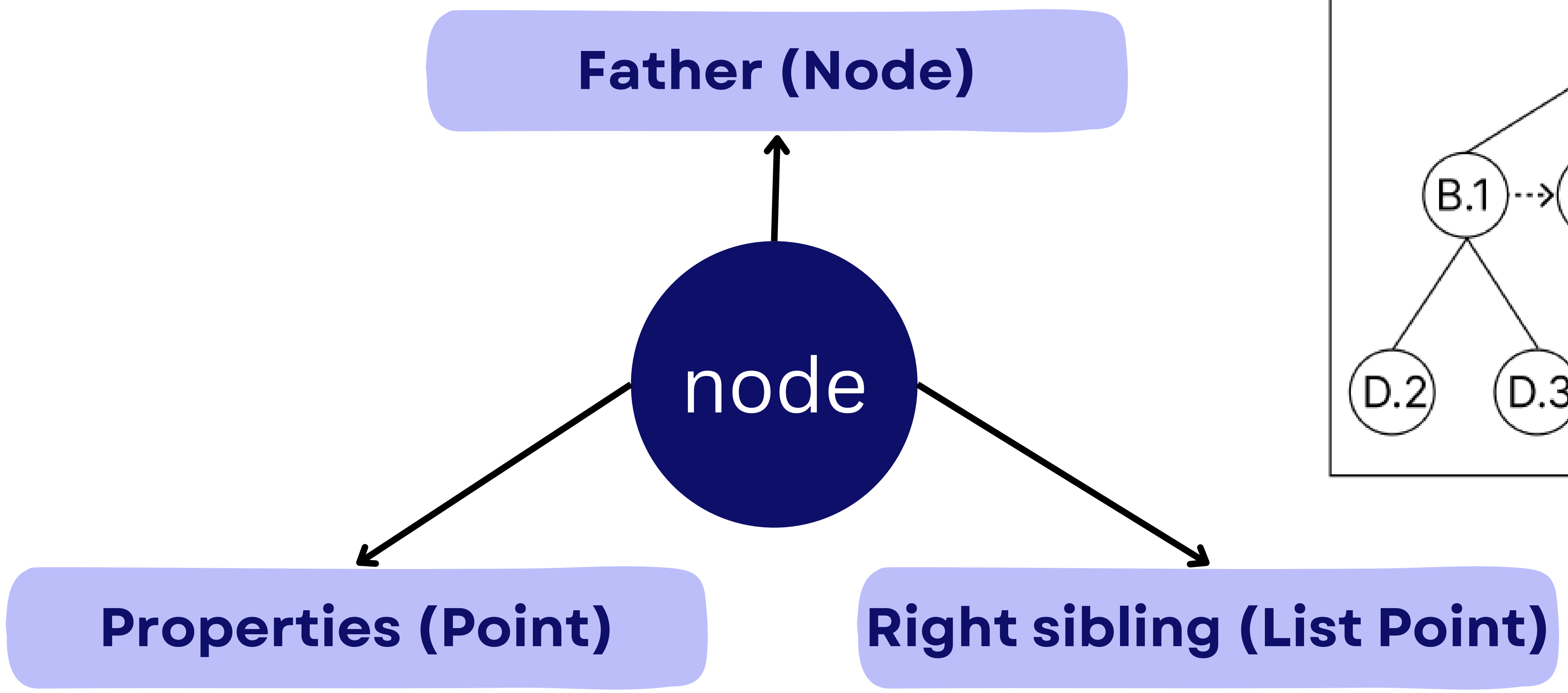
Big neighborhood list with k = 1

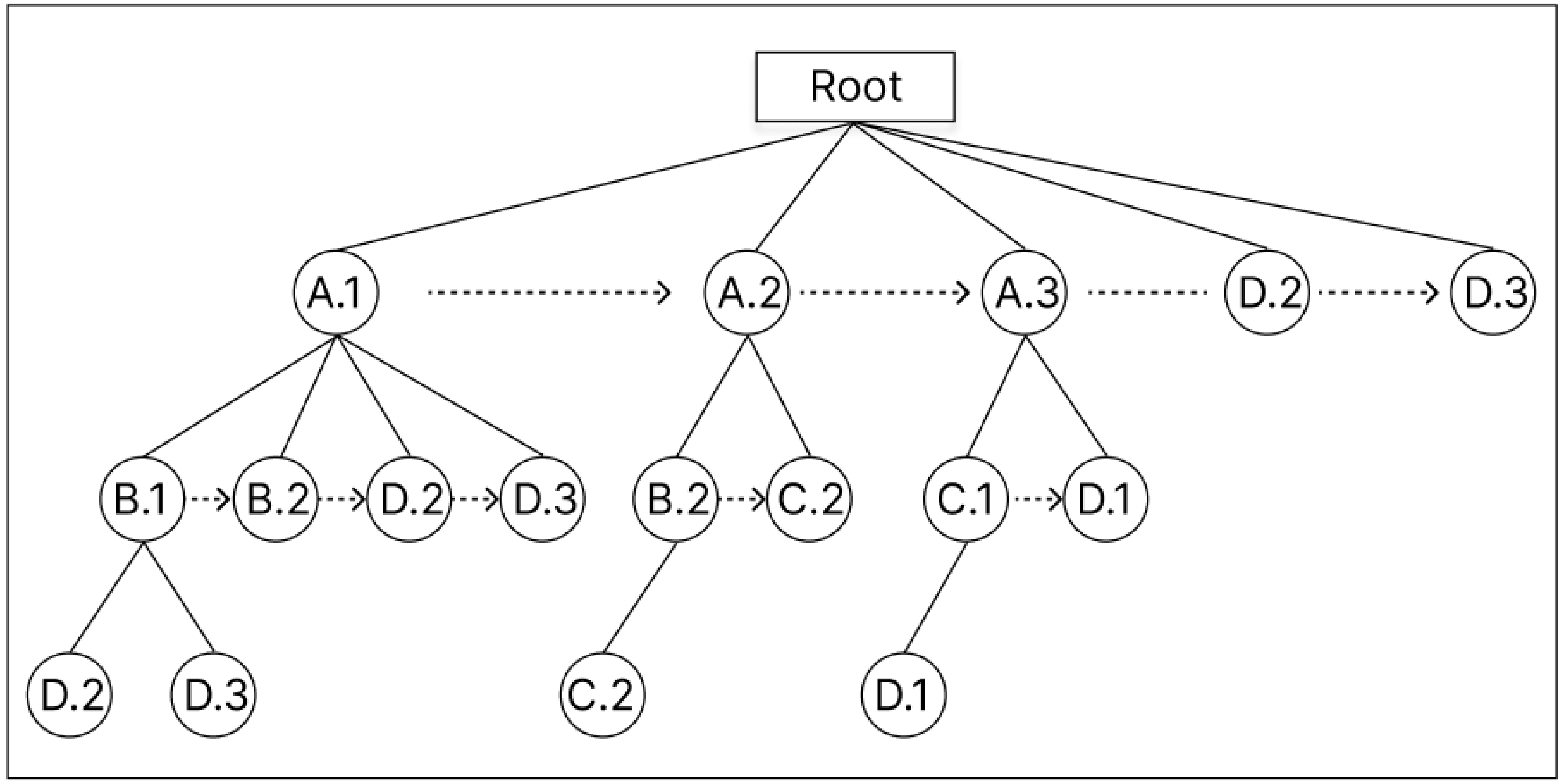
Instance	BNs	Instance	BNs	Instance	BNs	Instance	BNs
A.1	B.1, B.2, C.1, D.2, D.3	B.1	C.1, D.1, D.2, D.3	C.1	D.1, D.2, D.3	D.1	-
A.2	B.3, C.2	B.2	C.1, D.2, D.3	C.2	-	D.2	-
A.3	B.1, B.2, C.1, D.2, D.3	B.3	C.2	C.3	-	D.3	-

Big neighborhood list with k = 2

Clique approach: DFCIS

A Tree structure have bean build for finding cliques.



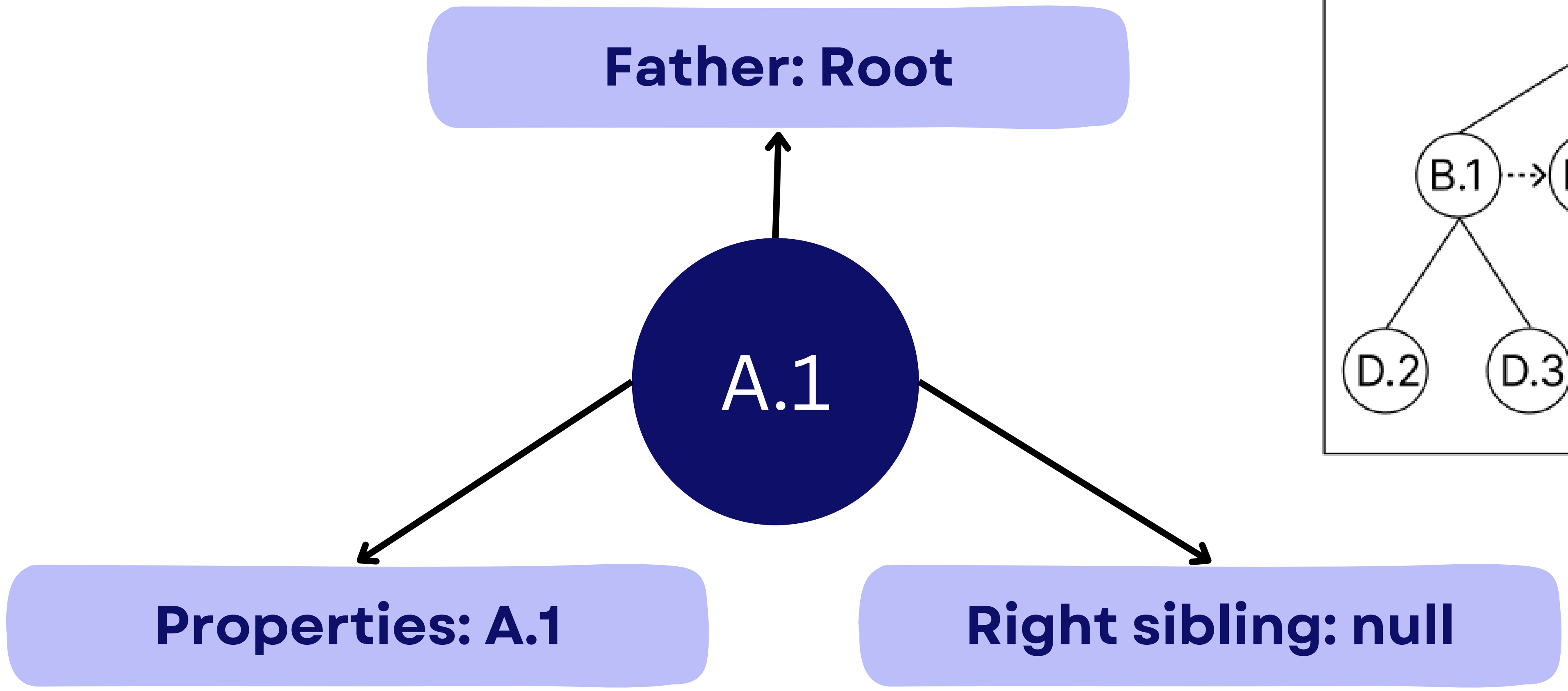


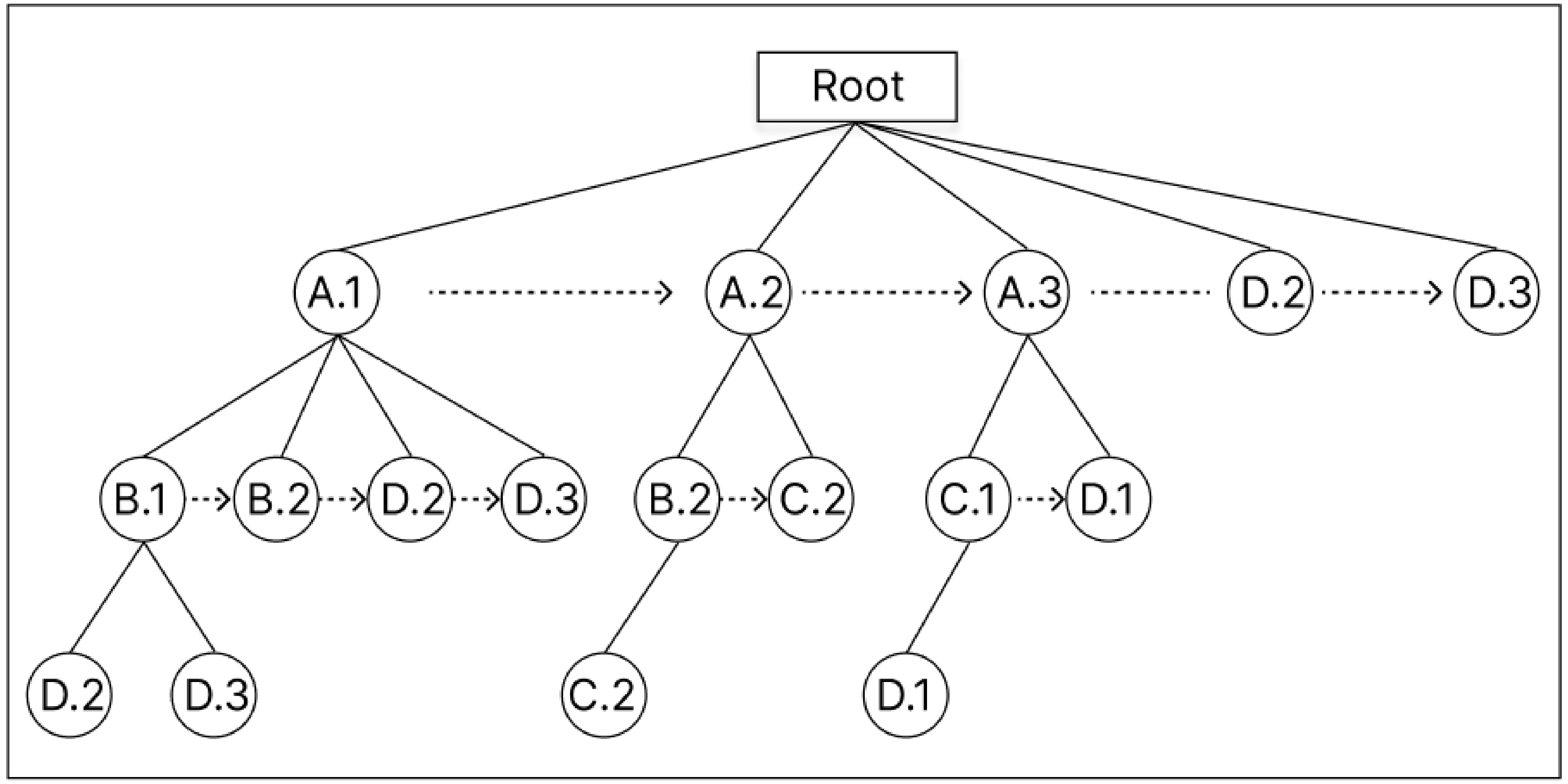
Example of I-Tree

Clique approach: DFCIS

There are 2 type of node:

Head node

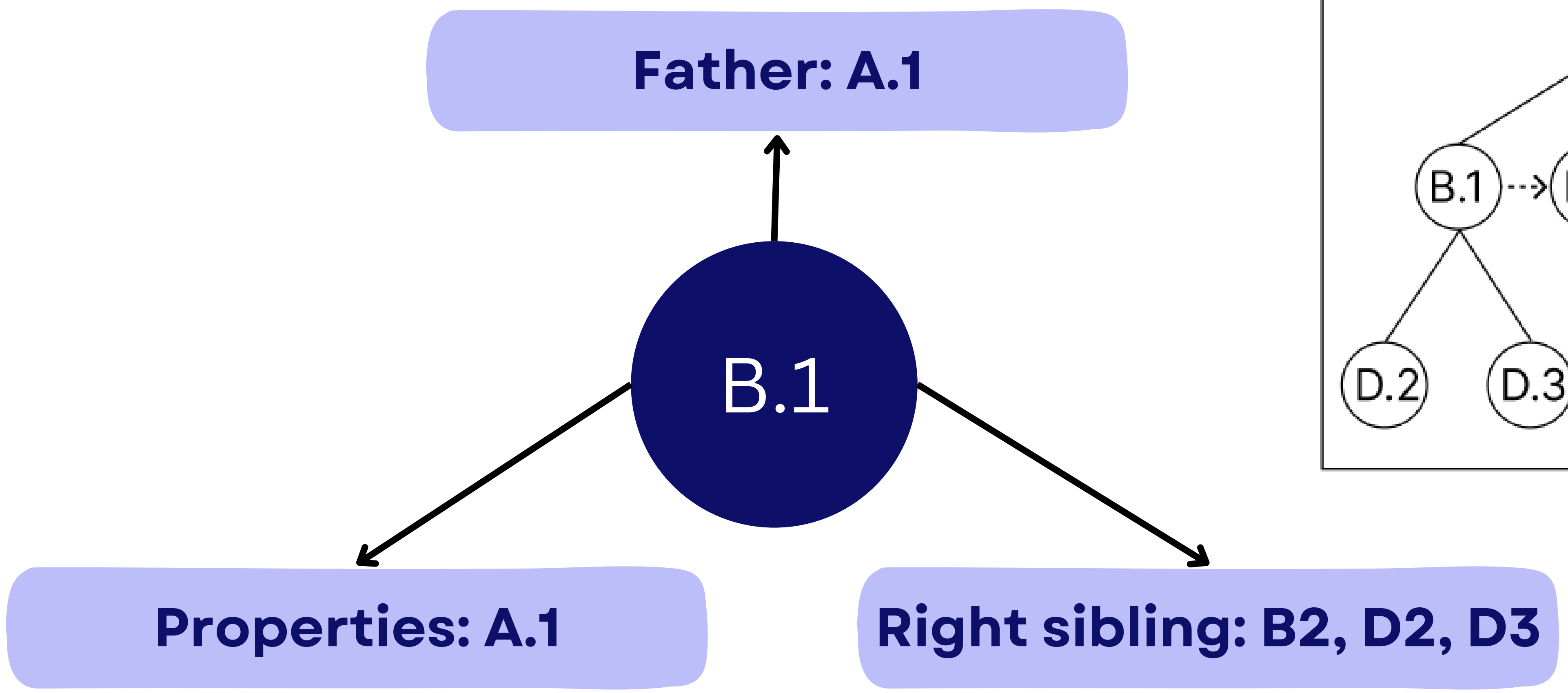


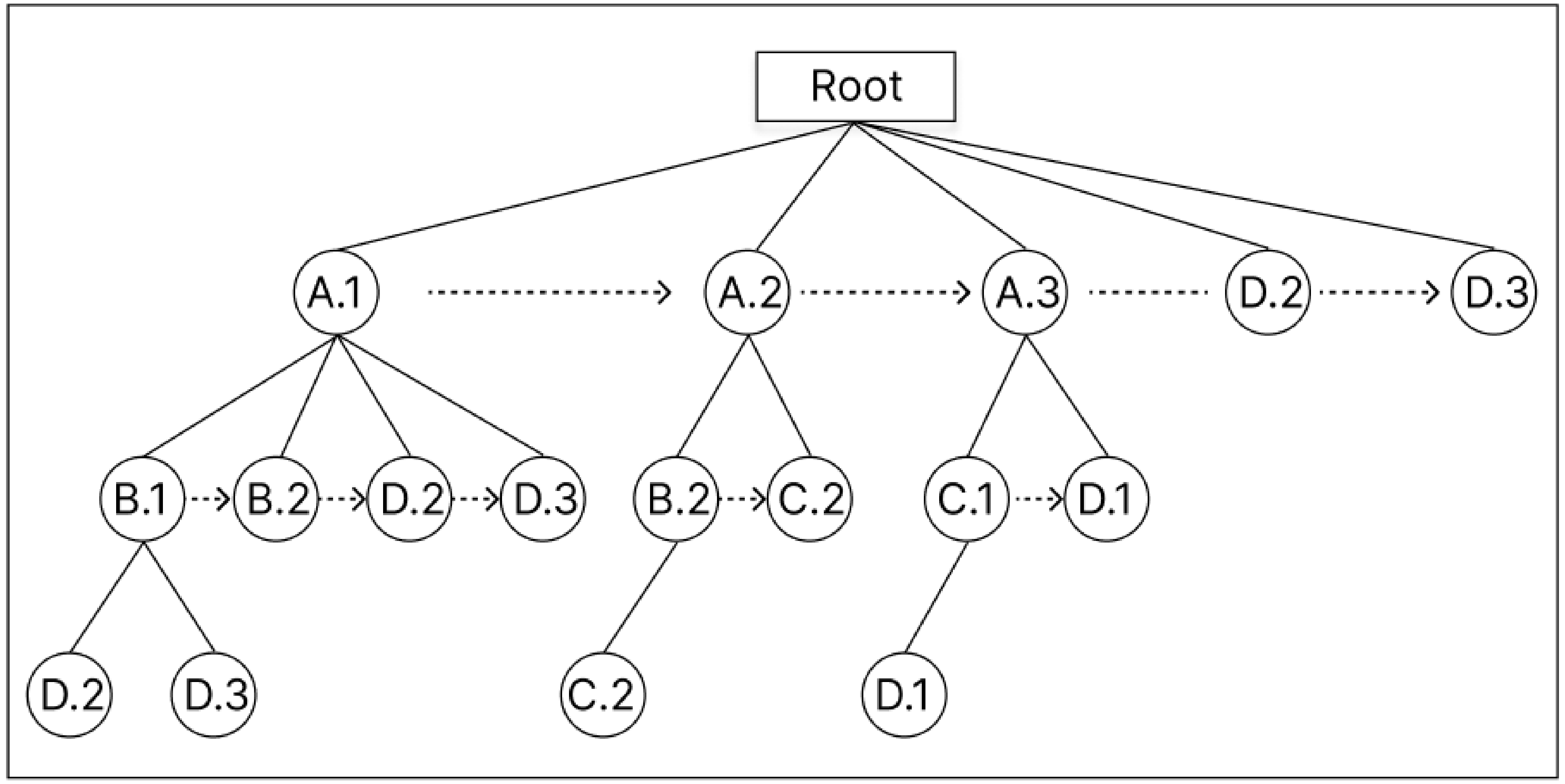


Clique approach: DFCIS

There are 2 type of node:

Other node



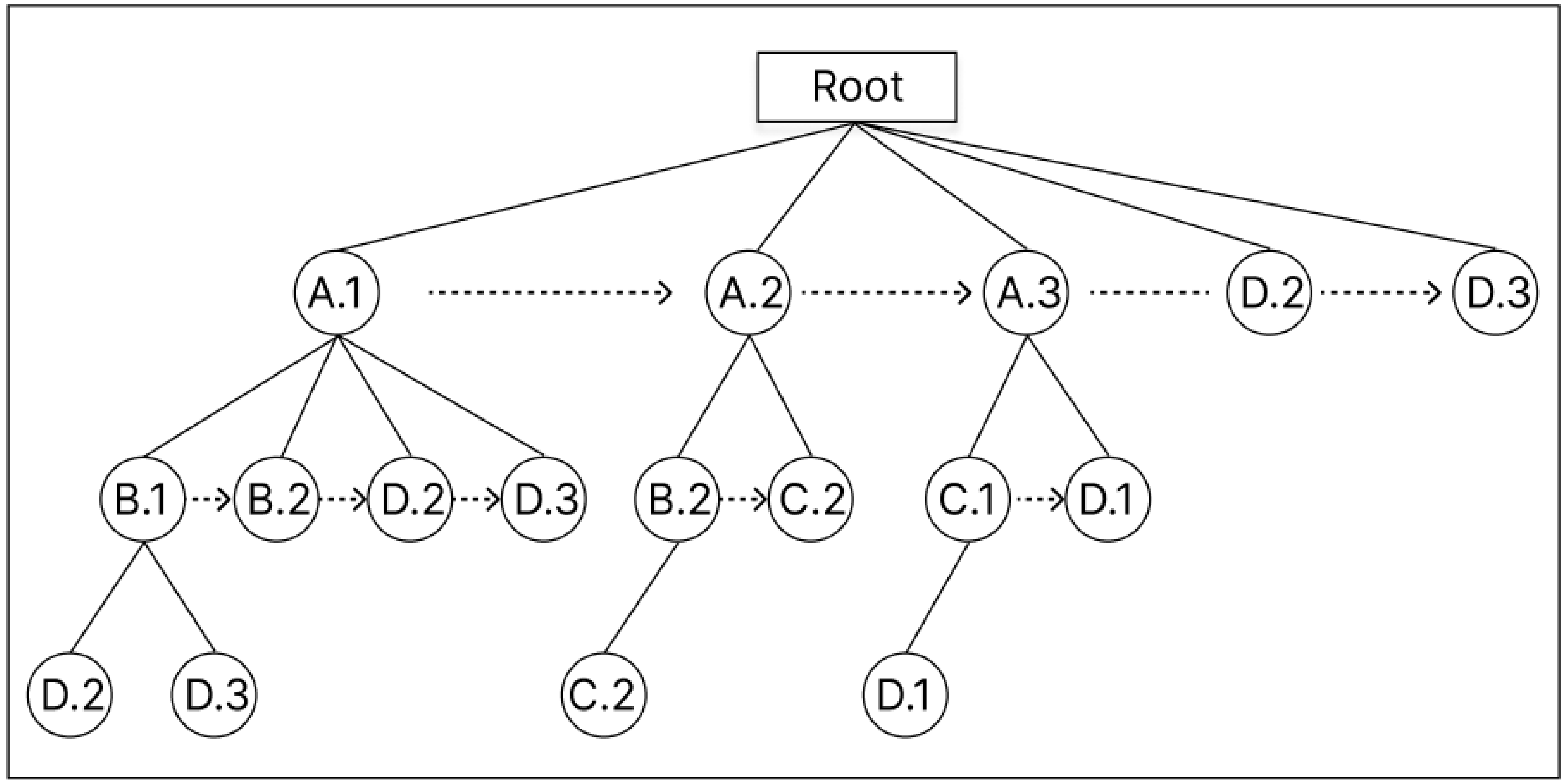


Clique approach: DFCIS

When traveling to a node that has no children, that node is called a leaf node. Trace back to root can get a clique.

Example:

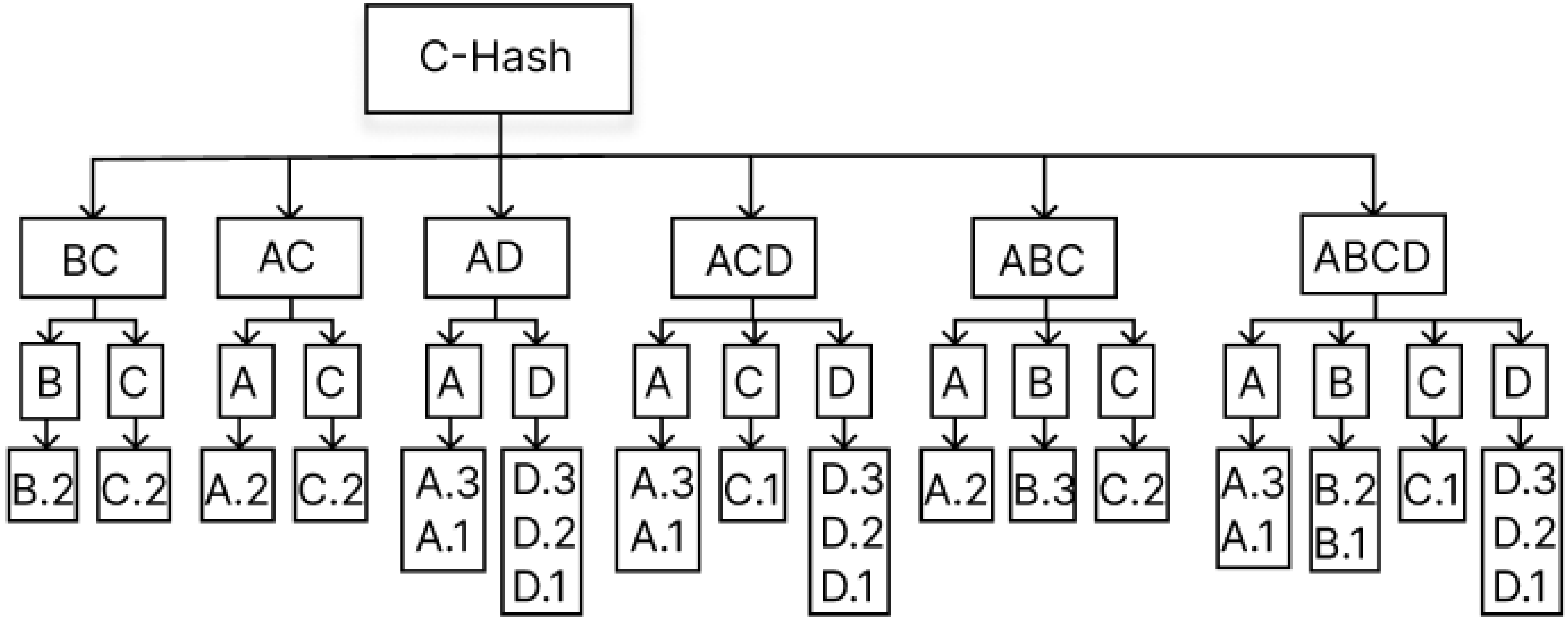
- **Node D.2:**
  - Trace back: D.2 -> B.1 -> A.1
  - Clique: A.1 , B.1, D.2
- **Node C.2:**
  - Trace back: C.2 -> B.2 -> A.2
  - Clique: A.2 , B.2, C.2



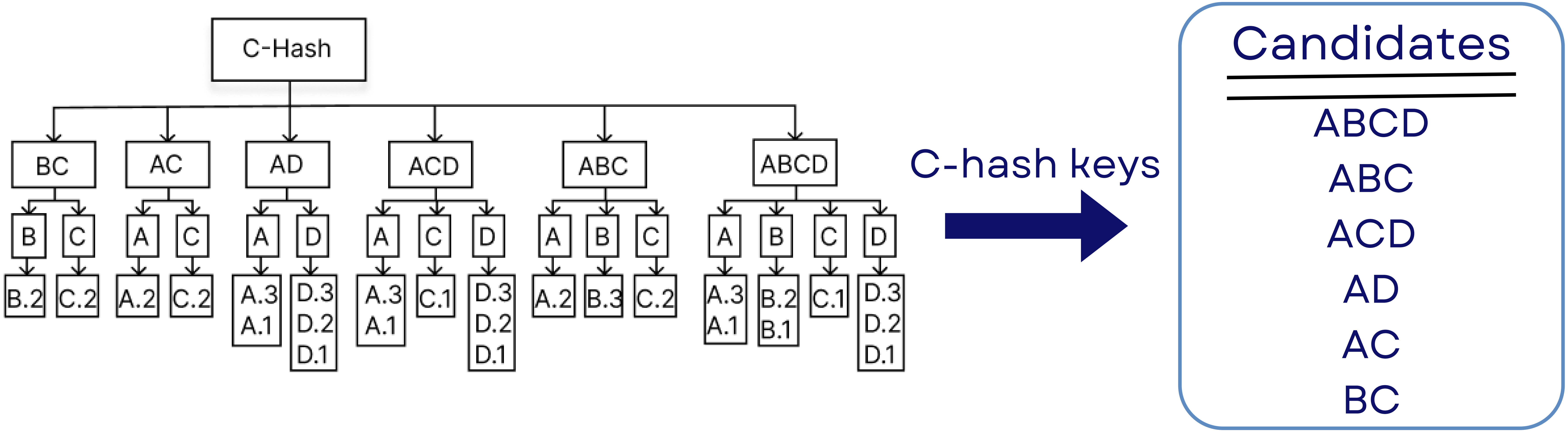
Candidate generation

Cliques			
B.3	C.2		
A.3	D.1		
A.3	D.2		
A.3	D.3		
A.2	C.2		
A.1	D.2		
A.1	D.3		
A.3	C.1	D.1	
A.3	C.1	D.2	
A.3	C.1	D.3	
A.2	B.3	C.2	
A.1	C.1	D.2	
A.1	C.1	D.3	
A.3	B.1	C.1	D.1
A.3	B.1	C.1	D.2
A.3	B.1	C.1	D.3
A.1	B.1	C.1	D.2
A.1	B.1	C.1	D.3
A.1	B.2	C.1	D.2
A.1	B.2	C.1	D.3

Compressed clique hash (C-hash): efficiently data structure to stores and organizes information by grouping features and associating them with their corresponding instances



Prevalent co-location filtering



With each candidate, we can collect it's row-instances from Chash table to caculate it's PI.

Example: Candidate: BC

- SuperSet: BC, ABC, ABCD
  - B: B.2, B.3, B.1
  - C: C.2, C.1

Prevalent co-location filtering

PI(ABC) > min_prev

→ **All subsets include is prevalent colocation**

Candidates

~~ABCD~~

~~ABC~~

ACD

AD

~~AC~~

~~BC~~

Prevalent co-location filtering

PI(ABC) < min_prev

→ Add all subset of ABC to Candidates for caculate

- Subset(ABC): AB, AC, BC

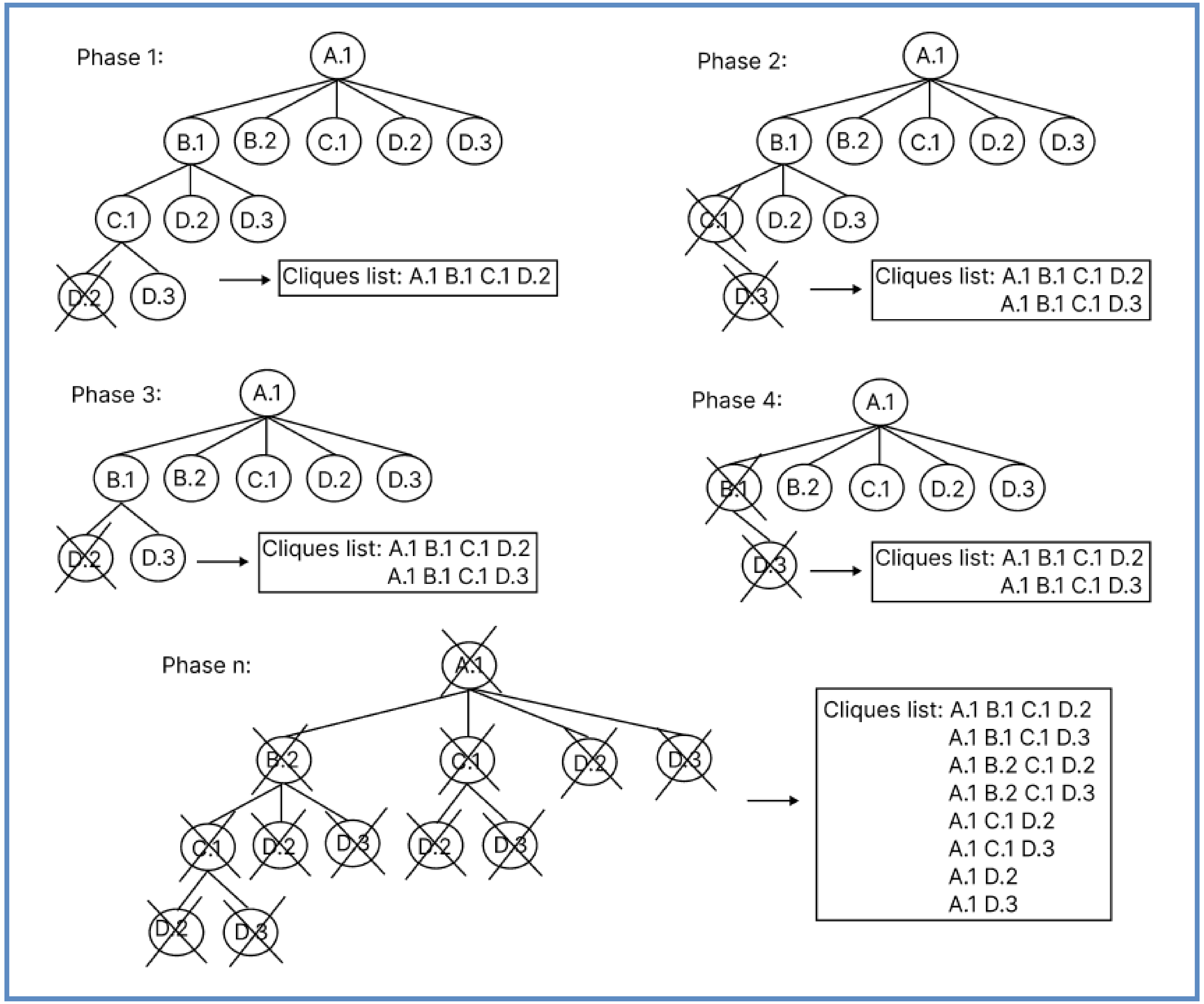
Candidates
~~ABCD~~
~~ABC~~
ACD
AD
AC
BC
AB

Clique approach: DFCIS

For efficient in caculate PI and create Tree

Includes:

- ClearNode
- Can Clique
- Old Instances list



EXPERIMENTAL RESULTS AND ANALYSIS

Experiment Setting

01 Choose Joinless, CP-tree-based (named Condense), and Delaunay triangulation-based co-location mining (DTC) to compare and perform on a Laptop with Intel(R) Core (TM) i7-8550u CPU@1.8 - 4.0 GHz and 16 GB main memory

02 Datasets: Four real datasets that are collected from points of interest in Beijing, China [2], Las Vegas, Toronto, USA2, and United Kingdom (UK)3, were used in our experiments. Moreover, two synthetic datasets were also used in our experiments

Experiment Setting

Name	Area	#feature	#instances	Distribution
Beijing	135km x 224km	17	90,257	Centralized + dense
Las Vegas	38km x 63km	19	31,592	Sparse + dense
Toronto	23km x 56km	19	20,309	Sparse + dense
UK	12,84km x 13,867km	26	143,621	Sparse + dense
Figure 8a, 9	5000km x 5000km	15	*	Dense

Experiment Setting

1 Compare the mining performance

The first experiment compares the mining performance of the four algorithms including running time and memory consumption based on the variations of two parameters: the minimum distance threshold (only for Joinless and Condense) and the minimum prevalence threshold.

2 Evaluate the scalability of DTkC

The second experiment compares the mining performance of the 2 algorithms including running time and memory consumption based on the variations the parameters: the number of instances and the minimum prevalence threshold (only for DTkC algorithm)

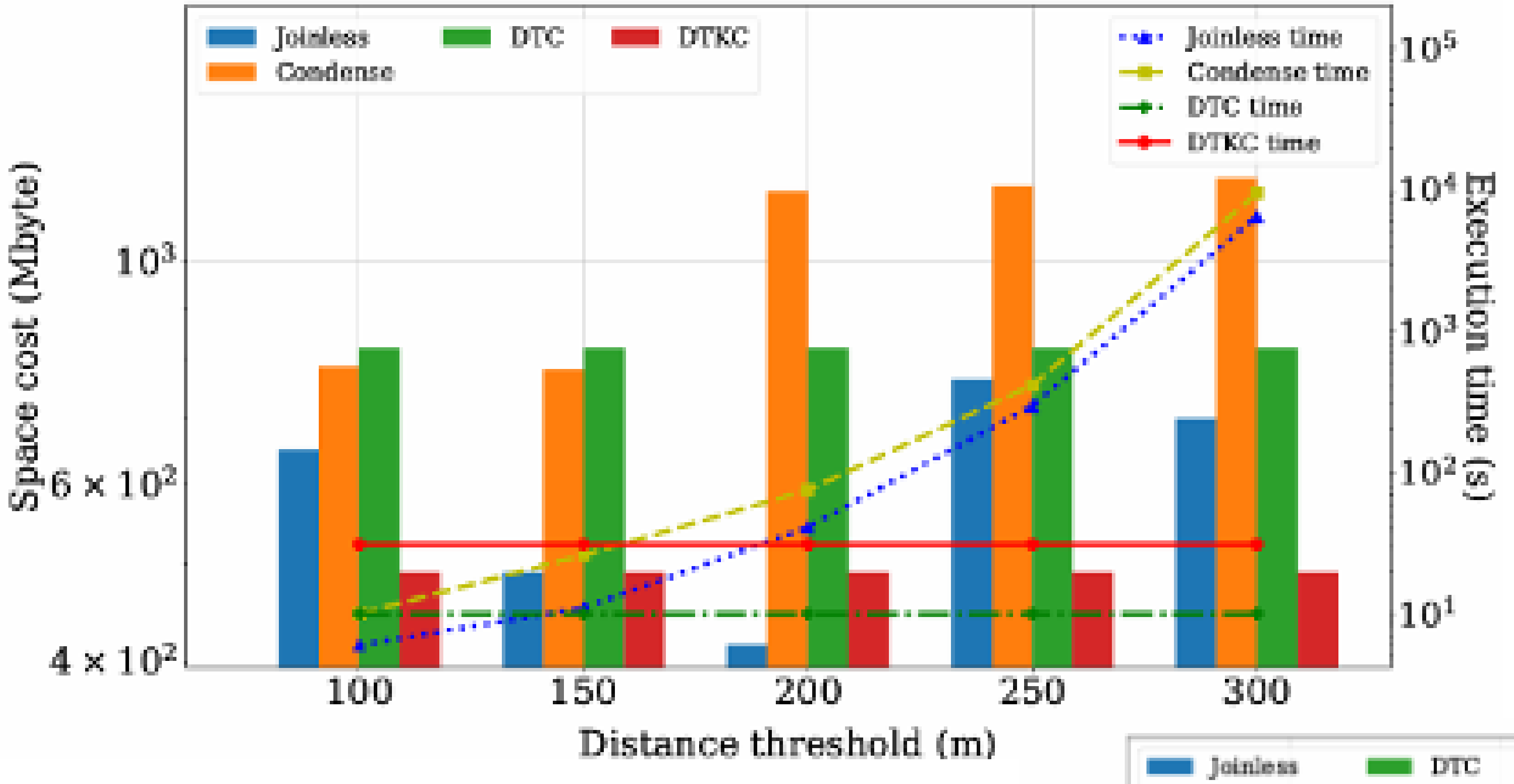
Compare the mining performance on the variations of the minimum distance threshold

1. For Joinless and Condensed algorithm, their running time will increase multiplicatively as distance thresholds increase. And a significant increase in memory consumption.

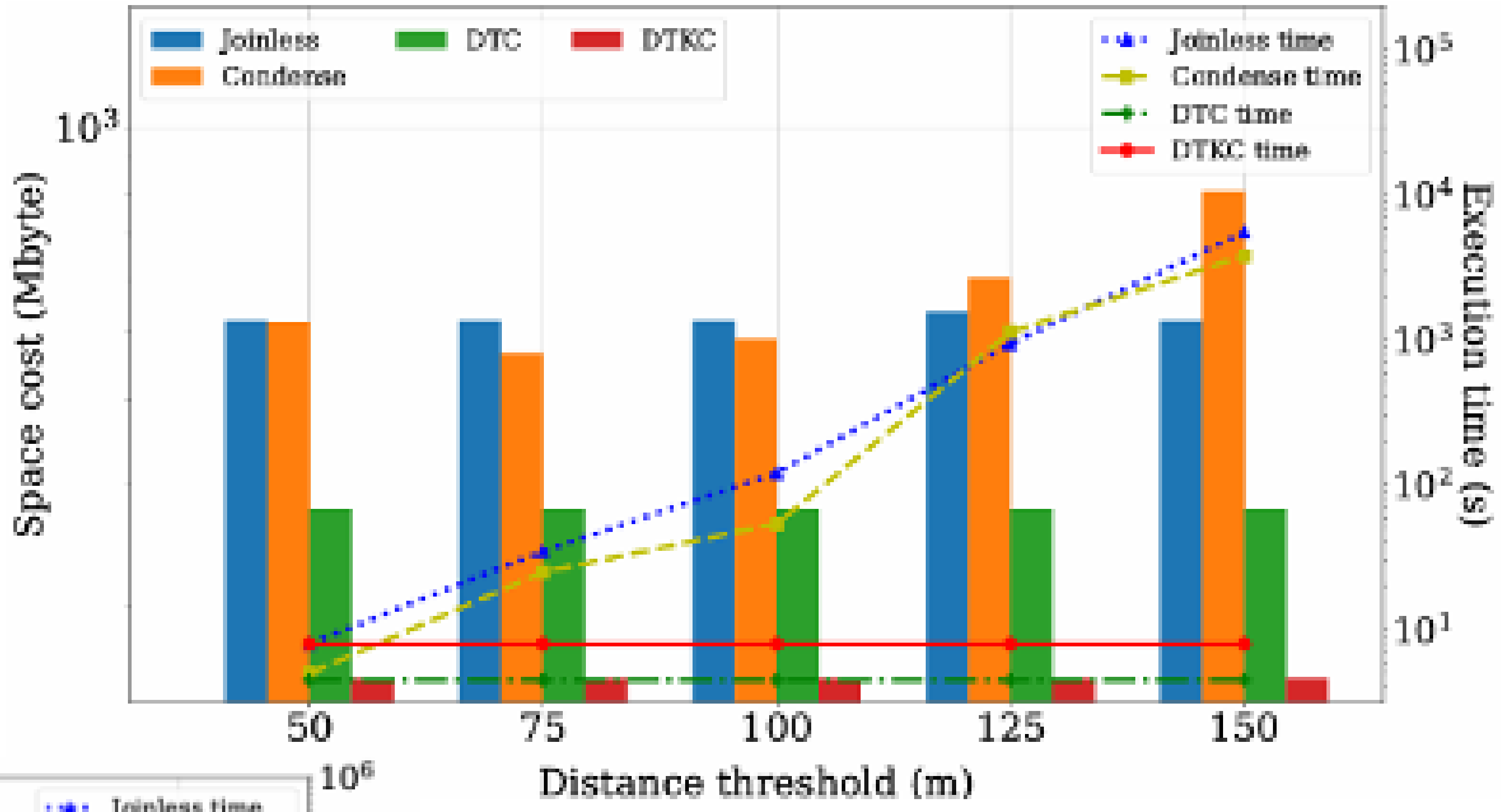
2. The DTC and DTkC algorithm do not change their running time and the memory consumption is always stable and lower compared to the aforementioned algorithms.

3. Beside that, The DTC algorithm ‘s execution time will be less than DTkC algorithm. However, DTkC exhibits significantly lower memory consumption than the DTC algorithm.

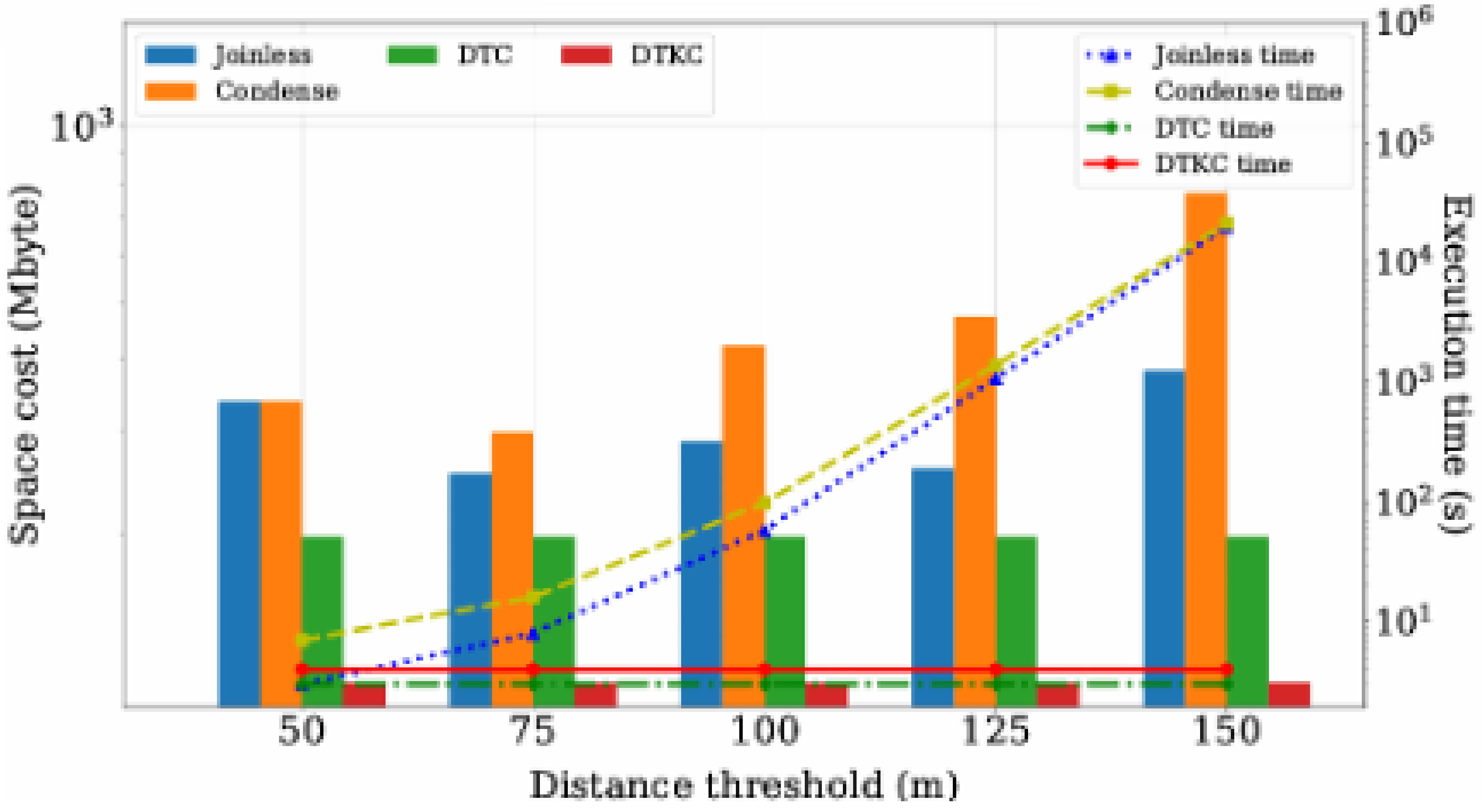
Compare the mining performance on the variations of the minimum distance threshold



(a) Beijing



(b) Las Vegas



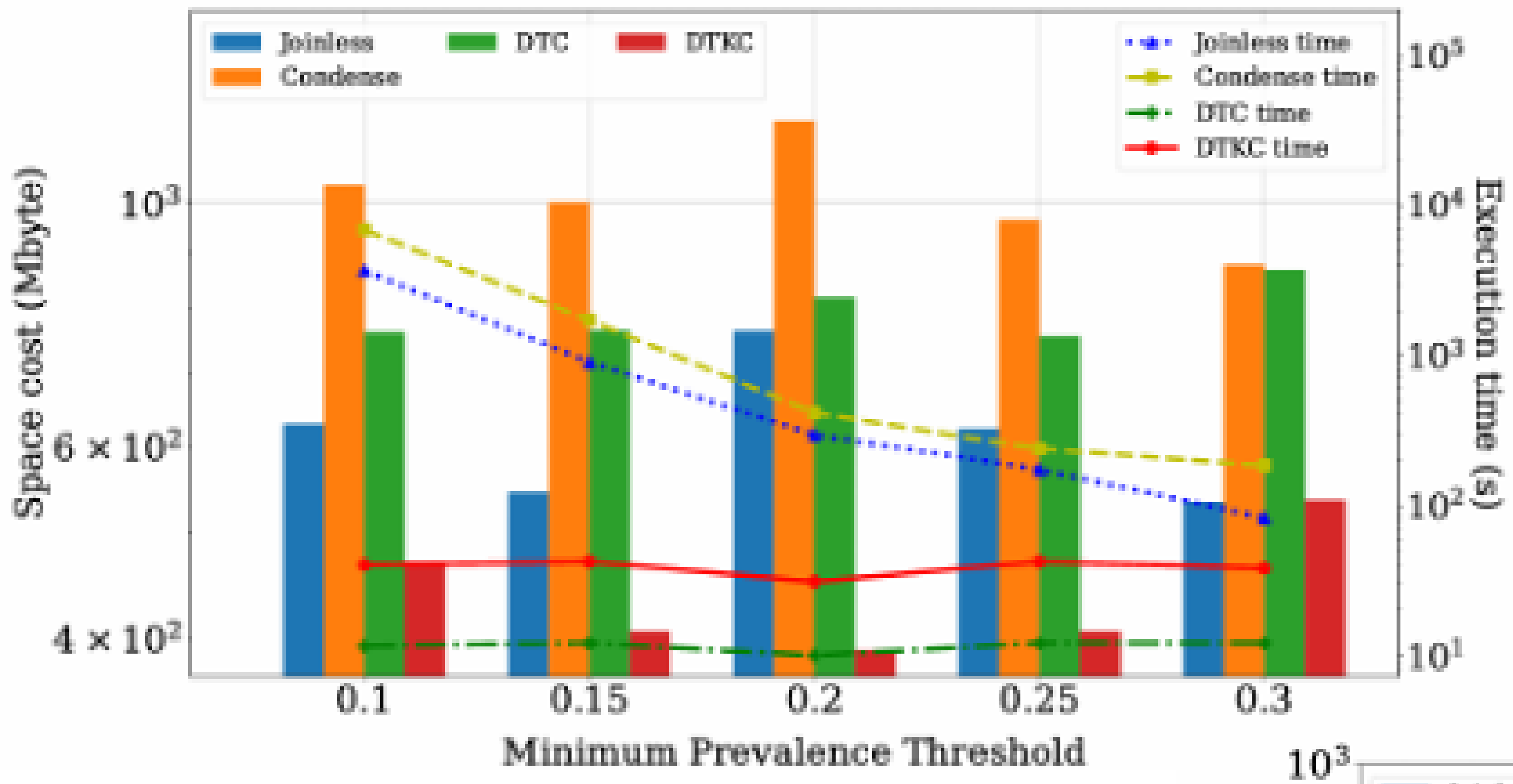
(c) Toronto

Compare the mining performance on the variations of the minimum prevalence threshold

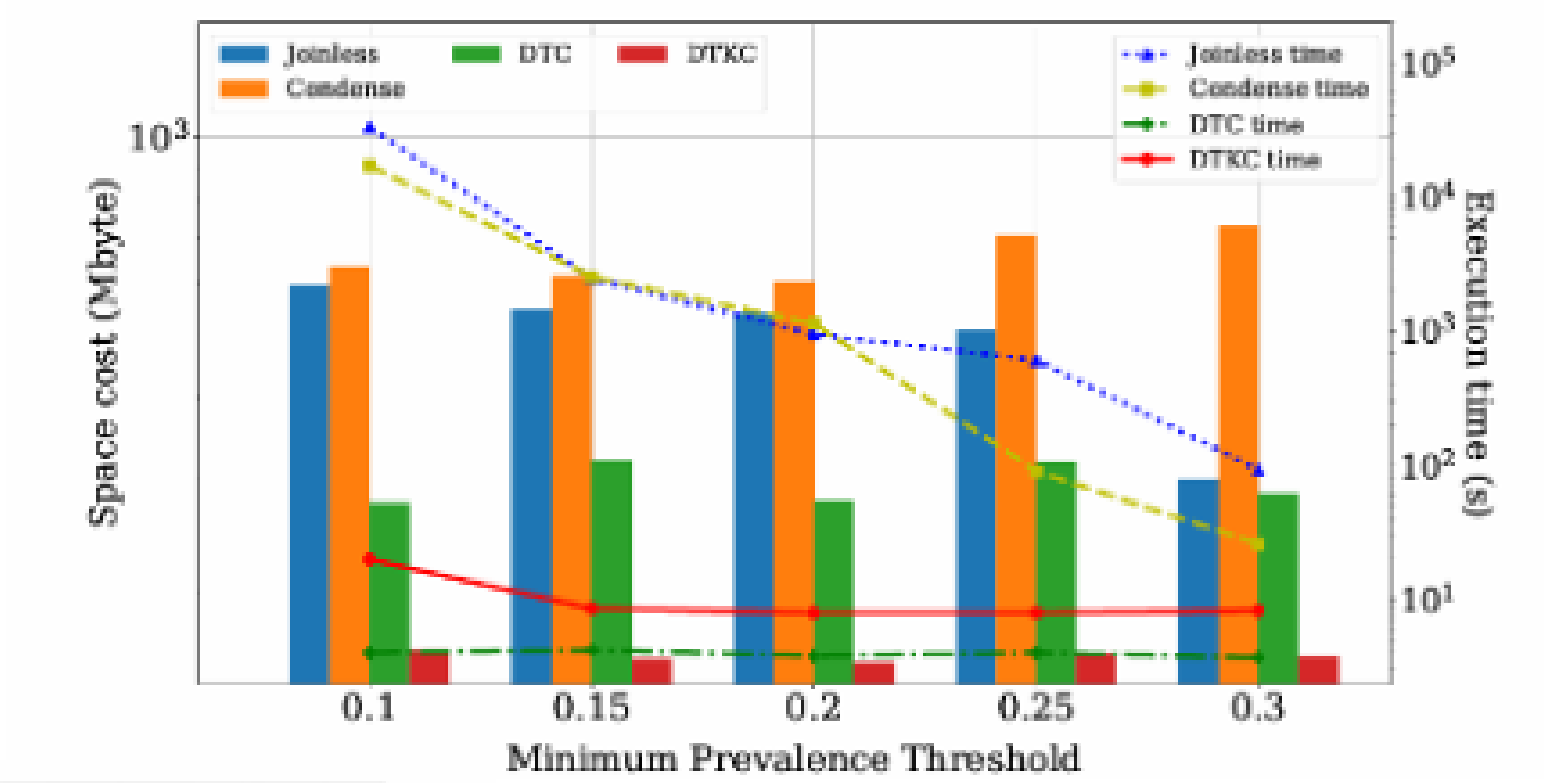
- With a smaller prevalence threshold (e.g., 0.1), the computation times of Joinless and Condense are significantly large. Although the computation times decrease when the prevalence threshold is increased, they still remain considerably high compared to DTC and DTkC. The memory consumption of Joinless and Condense decreases to some extent when the prevalence threshold increases.

- Changing the prevalence threshold does not significantly impact memory consumption on DTC and DTkC algorithm. However, DTkC still exhibits significantly lower memory consumption.

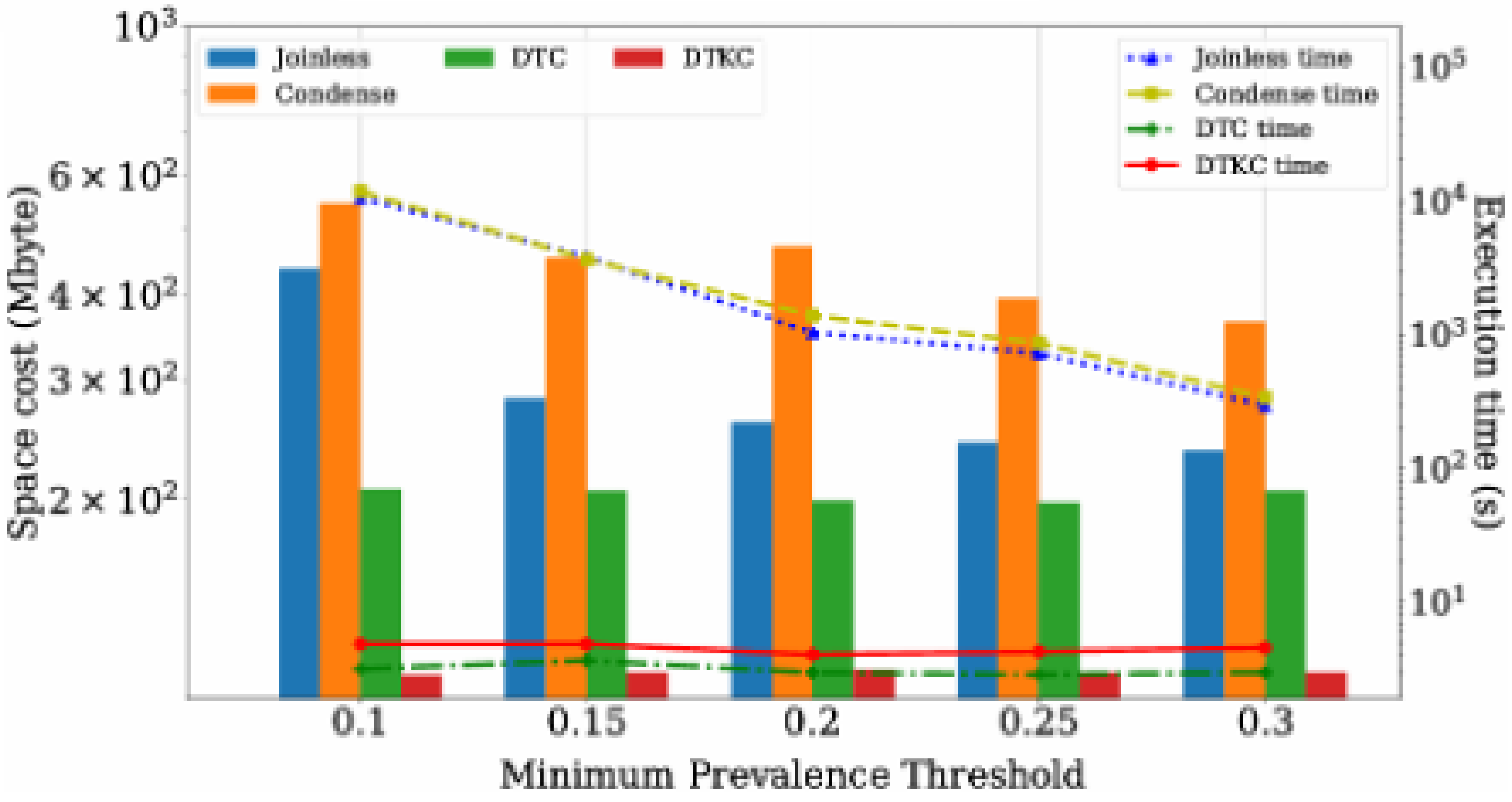
Compare the mining performance on the variations of the minimum prevalence threshold



(a) Beijing d = 250m

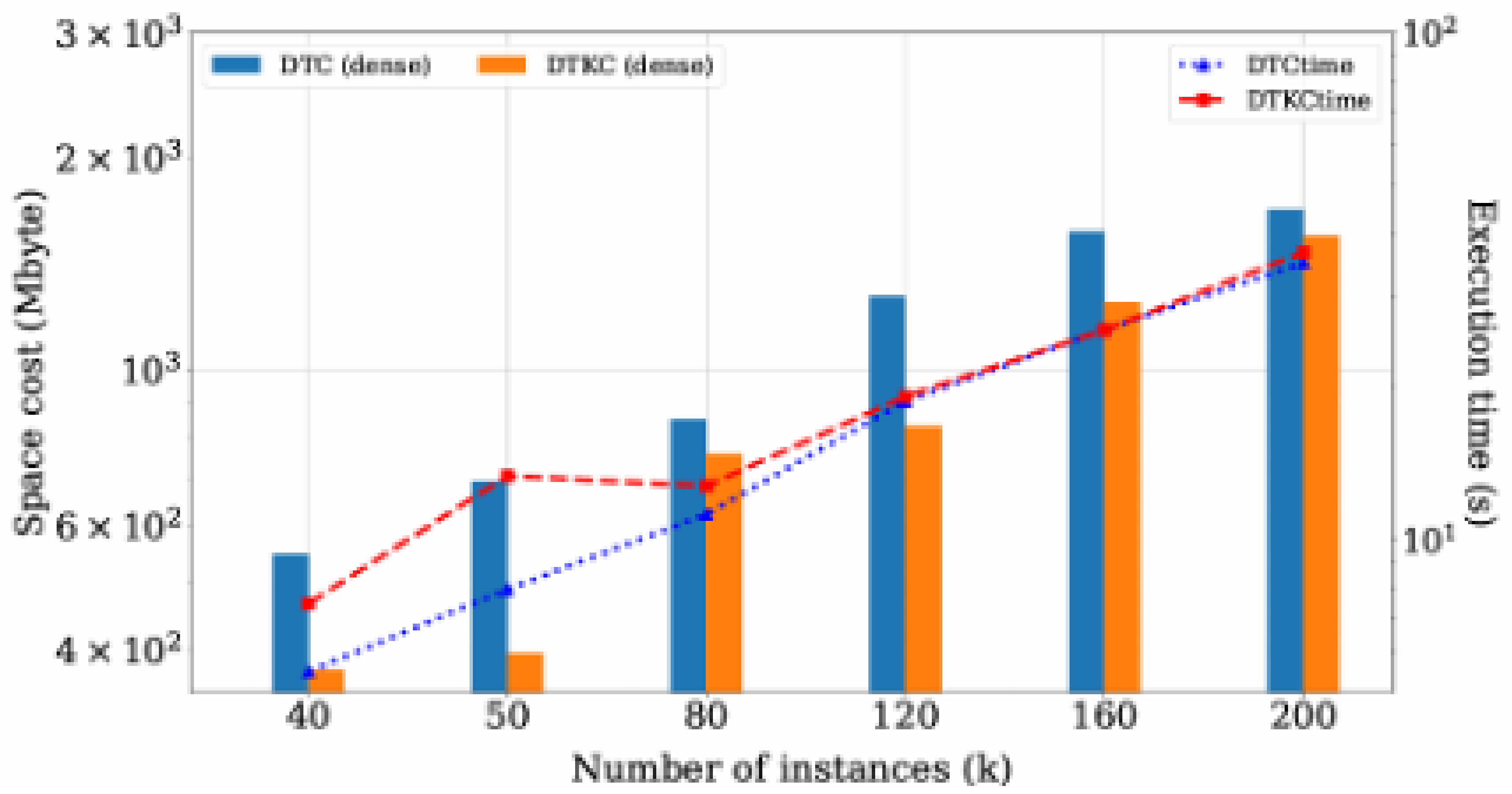


(b) Las Vegas d = 125m

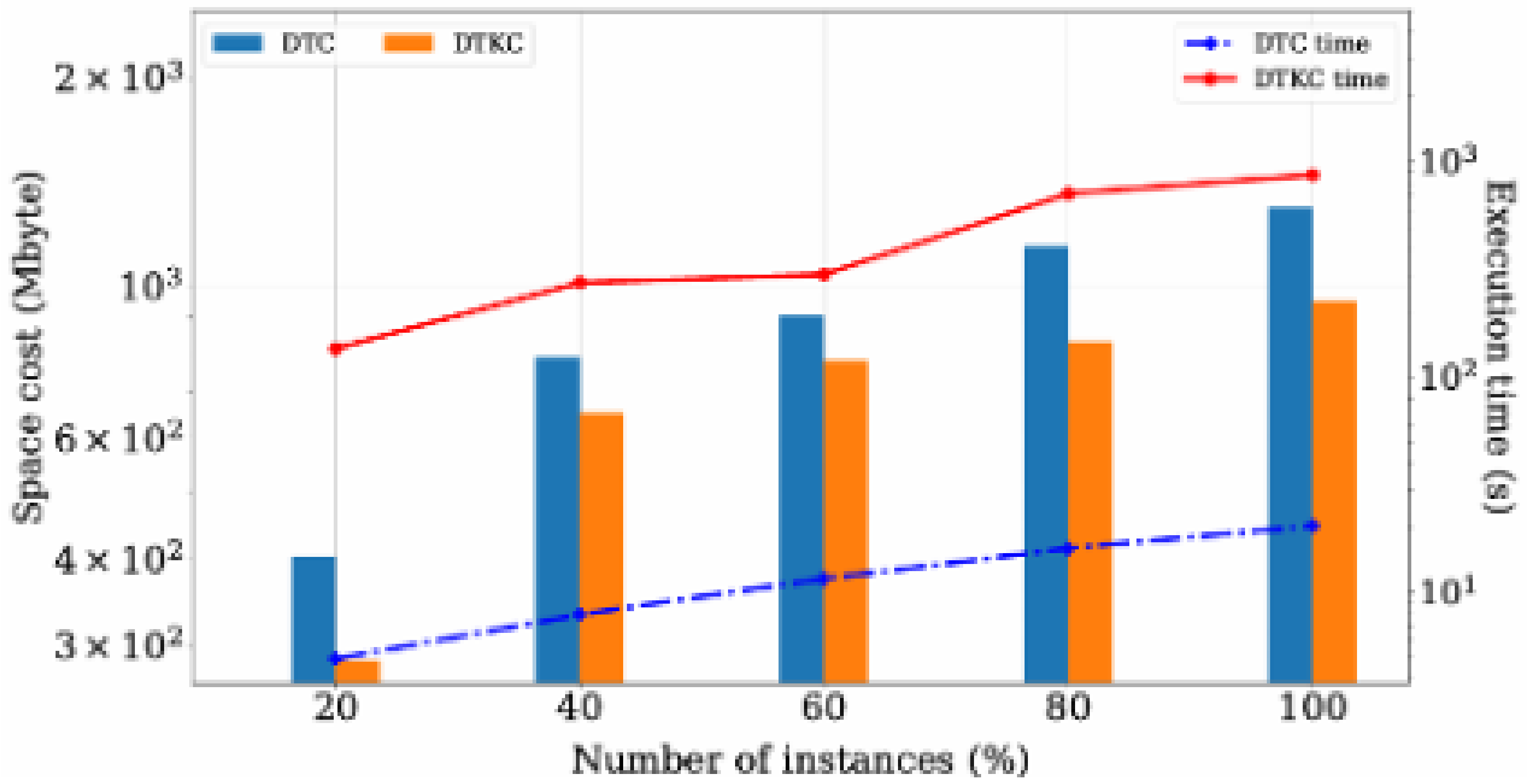


(c) Toronto d = 125m

Evaluate the scalability of DTkC on different numbers of instances



(a) Synthetic dataset

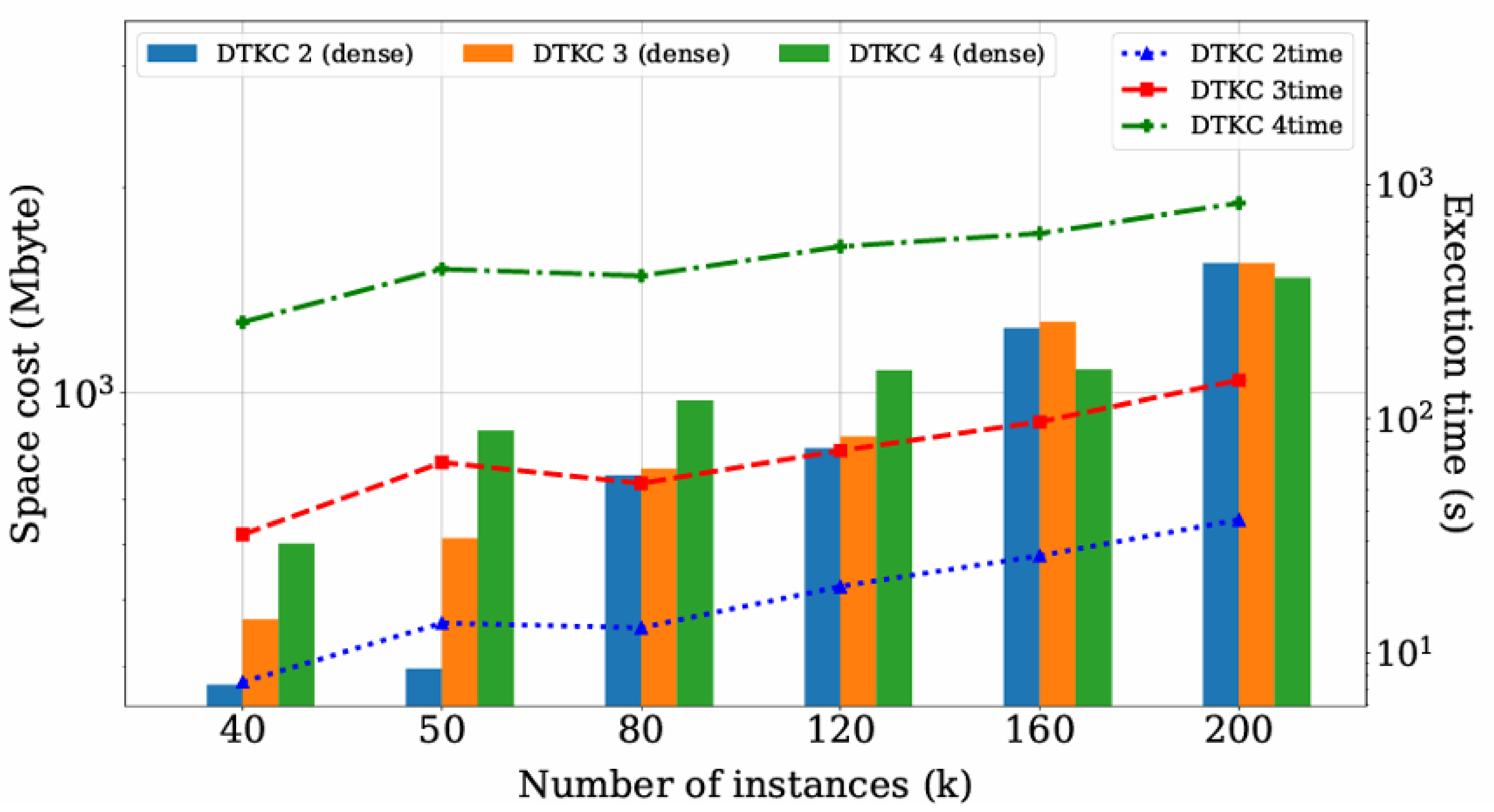


(b) UK dataset

In Figure a, the dataset is dense and the number of instances is large, the computation time for both the merging step and the depth-first clique search increases significantly. However, in Figure b, only the computation time for DTkC increases rapidly, while the computation time for DTC increases at a slower pace.

Evaluate the scalability of DTkC on different values of k

Both the execution time and space cost of DTkC show an increasing trend. However, the increase in space cost is not significant compared to the rate of increase in execution time



Conclusion

01 Mining PSCPs based on a distance threshold is challenging for users as it often leads to either missing or excessive patterns that may not align with their research objectives because it is difficult to find a suitable value of the threshold.

02 This work proposes a combined algorithm called DTkC, which leverages a Delaunay triangulation-based approach, incorporates the concept of k-order neighbors and uses a depth-first clique search strategy

03 In the future, we aim to improve the performance of the DTkC algorithm for datasets with moderate distribution density or datasets with a large number of features. We intend to explore and integrate suitable methods or techniques into the algorithm.

Question and Answer...

FPT University

Thank You