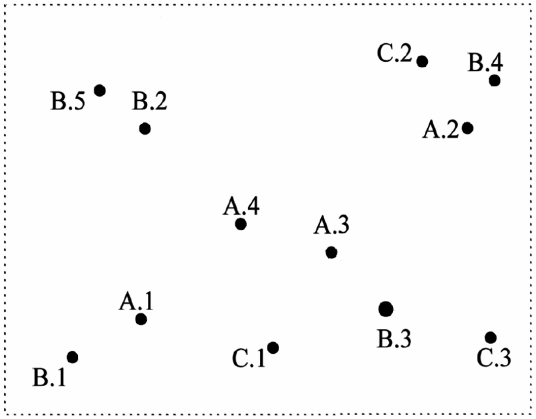
A Joinless Approach for Mining Spatial Colocation Patterns

Authors: J. Yoo, S. Shekhar

Presenter: Davin Wong
Spring 2007

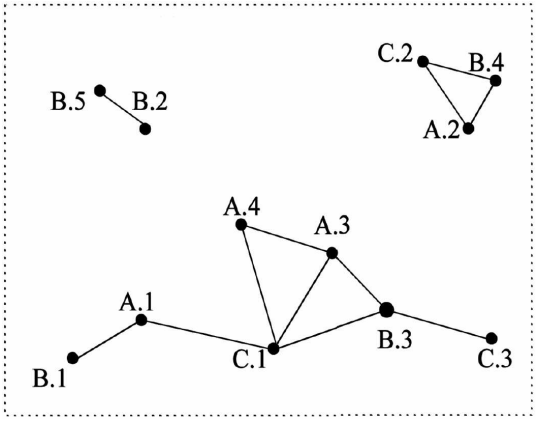
Spatial Dataset



Spatial Features:
A
B
C

Feature Instances:
A.1, A.2, A.3, A.4,
B.1, B.2, B.3, B.4, B.5,
C.1, C.2, C.3

Graph Representation



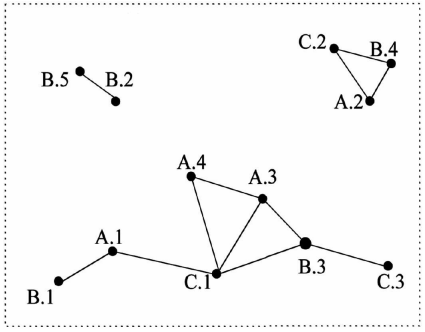
Given neighborhood distance d, draw an edge between two feature instances if their distance is ≤ d

Neighors:
B.2 – B.5
B.1 – A.1
A.1 – C.1
C.1 – A.4
...

Not Neighors:
A.1 – A.4
A.4 – B.3
...

Graph Representation

Clique in an undirected graph G is a set of vertices V such that for every two vertices in V, there exists an edge connecting the two.



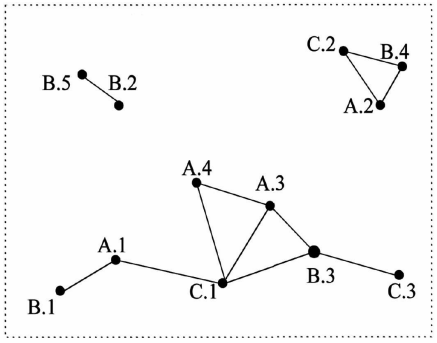
Cliques:
A.2 – B.4 – C.2
A.2 – B.4
B.2 – B.5
A.3 – A.4 – C.1
...

Not Cliques:
B.1 – C.1
A.3 – A.4 – B.3 – C.1
...

Note: All spatial feature instances in a clique are neighbors (≤ d distance)

Colocation Patterns

Colocation is a subset of spatial features, e.g. {A}, {B}, {C}, {A, B}, {A, C}, {B, C}, {A, B, C}



Colocation {A, B} Instances / Cliques:
A.1, B.1
A.3, B.3
A.2, B.4

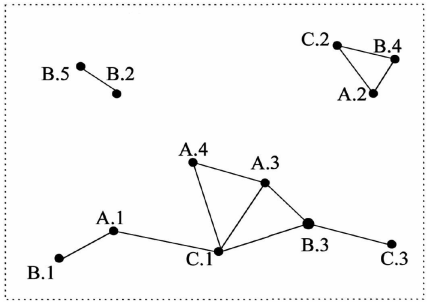
Colocation {B, C} Instances / Cliques:
B.3, C.1
B.3, C.3
B.4, C.2

Colocation {A, B, C} Instances / Cliques:
A.3, B.3, C.1
A.2, B.4, C.2

Note: Many colocation patterns are possible, we need a way to measure how interesting a colocation pattern is.

Colocation Interestingness – Participation Ratio

$$Pr(f_i, C) = \frac{\text{\# of distinct instances of feature } f_i \text{ in instances of colocation } C}{\text{\# of feature instances of } f_i}$$



Example: Pr(B, {B, C})

Colocation {B, C} Instances:
B.3, C.1
B.3, C.3
B.4, C.2

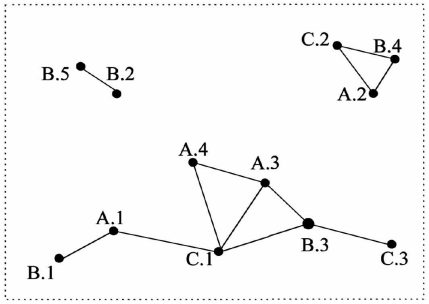
Feature B Instances:
{B.1, B.2, B.3, B.4, B.5,}

Hence, Pr(B, {B, C}) = 2/5

Note: Pr(A, {A}) = Pr(B, {B}) = Pr(C, {C}) = 1

Colocation Interestingness – Participation Index

$$Pi(C) = \text{prevalence}(C) = \min_{f_i} \{Pr(f_i, C)\}$$



Example: Pi({B, C})

Pr(B, {B, C}) = 2/5
Pr(C, {B, C}) = 3/3 = 1

Hence, Pi({B, C}) = 2/5

Example: Pi({A, B})

= min {Pr(A, {A, B}), Pr(B, {A, B})}
= min {3/4, 3/5}
= 3/5

Colocation Mining Algorithm

Input: F = set of spatial features, e.g. {A, B, C}
FI = set of spatial feature instances with coordinates, e.g. {A1, A.2, A3, A4, ...}
r = maximum neighbor distance
minPrev = minimum prevalent threshold
Output: PC = set of prevalent colocation patterns

Mine (F, FI, r, minPrev)
for k = 2 to |F|
C_k = find all candidate colocation patterns of size k
for each candidate colocation pattern P in C_k
CI = find all colocation instances of P
prev(P) = min {pr(f_1, P), pr(f_2, P), ..., pr(f_k, P)}, where f_i = a feature type in P
if prev(P) ≥ minPrev
PC = PC U {P}

these sets can be huge!

Colocation Mining Algorithm Optimization

Problem 1: Find all candidate colocation patterns of size k

Solution: Use the anti-monotone property of the prevalence measure:

$$prev(C_k) \leq prev(C_{k-1}), \quad \text{w.r.t. subset operator}$$

In other words, if colocation {A, B, C} is prevalent, then colocation {A, B}, {A, C} and {B, C} which are subsets of {A, B, C} are also prevalent.

Hence, we can use prevalent colocations of size k-1 to construct **candidate colocations** of size k.

Note: We still have to check whether the candidates are really prevalent i.e. they meet the minimum prevalent threshold.

Colocation Mining Algorithm Optimization

Problem 2: Find all colocation instances (cliques) of candidate colocation P

More precisely, how to find cliques efficiently from the spatial data?

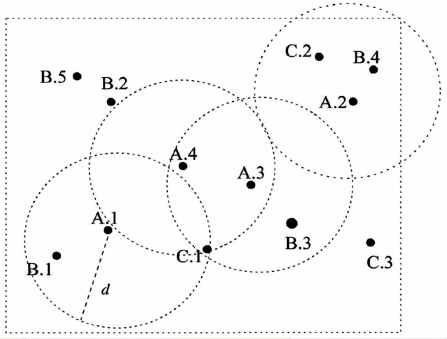
Solution: Use some kind of model representation to capture the neighbor relationship of the spatial data.

One possible choice is the **star neighborhood partition model**.

Star Neighborhood Partitioning

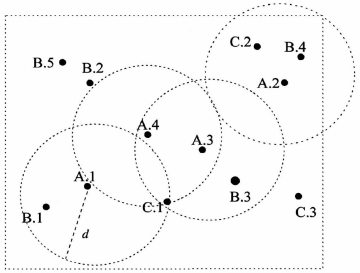
- **Star neighborhood** of a feature instance is:
  - a set consisting of the instance itself plus any other feature instances within the predefined neighbor distance.
  - the feature type of the neighbor instances must be greater than the feature type of the center instance in lexical order.

Star neighborhood area of A.1, A.2, A.3 and A.4 (dashed circles) ▶

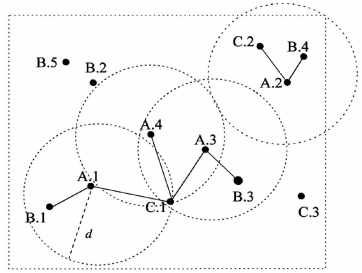


Star Neighborhood Partitioning

- **Star neighborhood** of a feature instance is:
  - a set consisting of the instance itself plus any other feature instances within the predefined neighbor distance.
  - the feature type of the neighbor instances must be greater than the feature type of the center instance in lexical order.



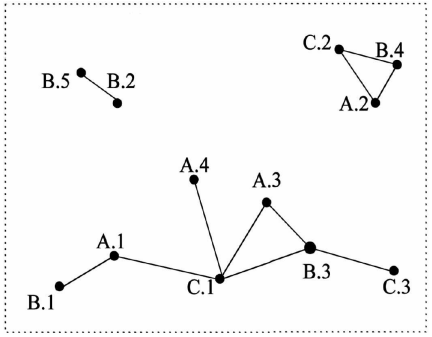
Star neighborhood area of A.1, A.2, A.3 and A.4 (dashed circles)



Star neighbors of A.1, A.2, A.3 and A.4 (edges)

Star Neighborhood Partitioning

Applying star neighborhood partitioning to our example...



=

Center feature	Center object	Neighbor objects
A	A.1	A.1 , B.1 , C.1
	A.2	A.2 , B.4 , C.2
	A.3	A.3 , B.3 , C.1
	A.4	A.4 , C.1
B	B.1	B.1
	B.2	B.2
	B.3	B.3 , C.1, C.3
	B.4	B.4 C.2
	B.5	B.5
C	C.1	C.1
	C.2	C.2
	C.3	C.3

Star Neighborhood – Advantage #1

- Candidate colocation instances (a.k.a. **star instances**) can be produced quickly.

Center feature	Center object	Neighbor objects
A	A.1	A.1 , B.1 , C.1
	A.2	A.2 , B.4 , C.2
	A.3	A.3 , B.3 , C.1
	A.4	A.4 , C.1
B	B.1	B.1
	B.2	B.2
	B.3	B.3 , C.1, C.3
	B.4	B.4 C.2
	B.5	B.5
C	C.1	C.1
	C.2	C.2
	C.3	C.3

} Candidate instances for {A}, {A, B}, {A, C}, {A, B, C}

} Candidate instances for {B}, {B, C}

} Candidate instances for {C}

Star Neighborhood – Advantage #2

- Candidate colocation can be coarsely filtered from the participation index of the star instances.

Example

Given candidate colocation {A, B, C}

A	B	C	
A.1	B.1	C.1	star instances
A.2	B.4	C.2	
A.3	B.3	C.1	
3/4	3/5	2/3	◄ Participation Ratio Estimates

If the estimated p.i. is less than minimum prevalent threshold, then discard the candidate.

- In general, Pi(star instances of C) ≥ Pi(C)

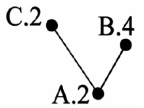
Star Neighborhood – Advantage #3

- Cliqueness of a star instance can be checked from the star neighborhoods.

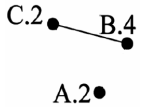
Example

Given star instance {A.2, B.4, C.2}

We know A.2's neighbors are B.4 and C.2. If {B.4, C.2} is also a star instance, then {A.2, B.4, C.2} is a clique.



Star neighborhood of A.2

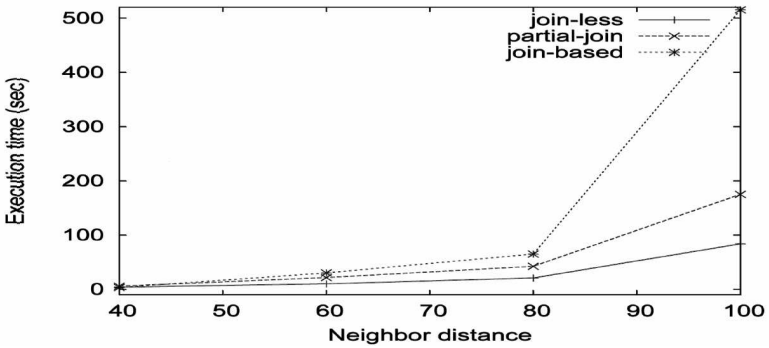


Star neighborhood of B.4

- In general, star instance $\{o_1, o_2, ..., o_k\}$ is a clique if subinstance $\{o_2, ..., o_k\}$ is a clique.

Performance Comparison

- The joinless approach which utilizes star neighborhood partitioning is more scalable.



~ The End ~