Knowledge Discovery in the Big Data era

Who are these gold prospectors?

- ORPAILLEUR project-team includes a large panel of experts from different domains:
  - Biologists
  - Chemists
  - Computer Scientists
  - Physician

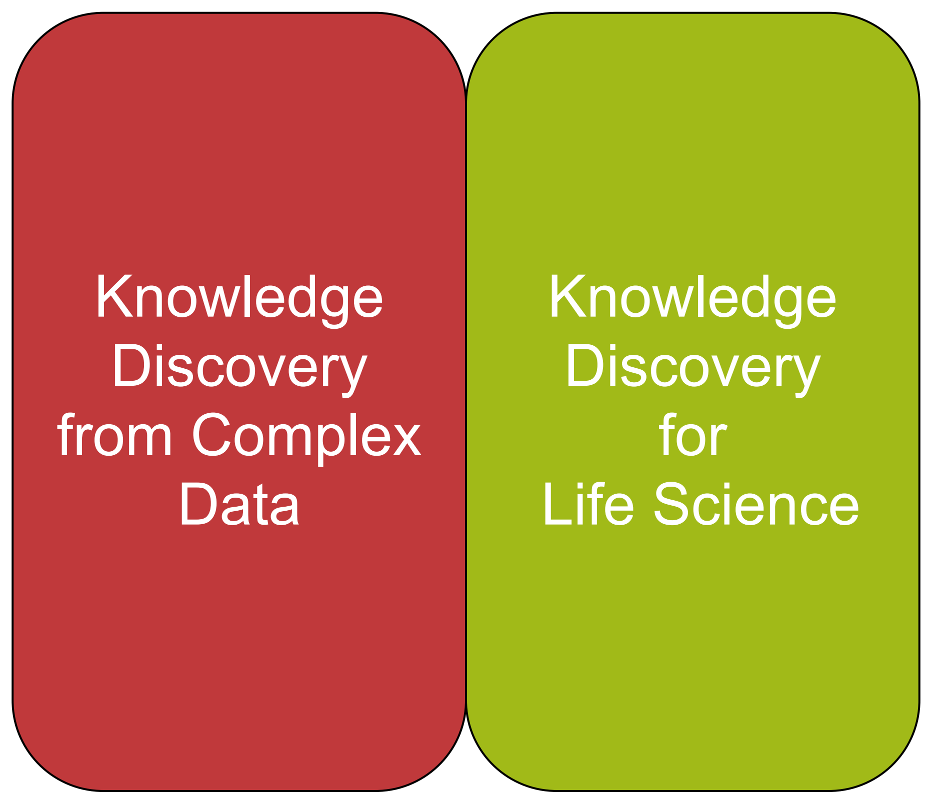
Who are these gold prospectors?

- ORPAILLEUR project-team includes a large panel of experts from different domains:
  - Biologists
  - Chemists
  - Computer Scientists
  - Physician

Knowledge Discovery
guided by
Domain Knowledge

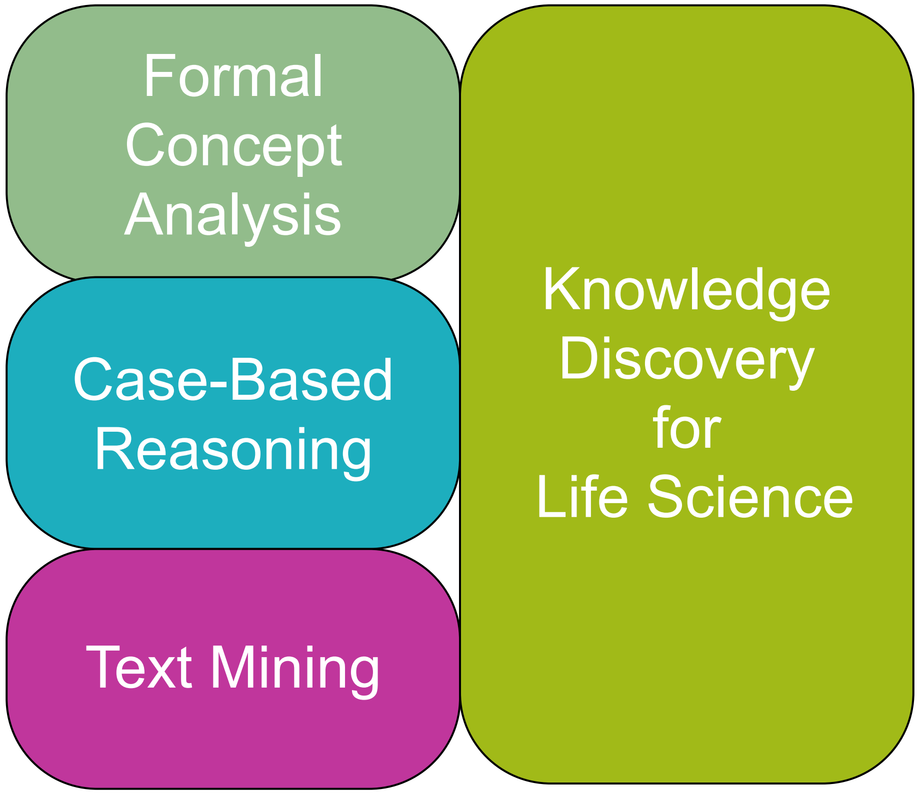
Who are these gold prospectors?

- ORPAILLEUR project-team includes a large panel of experts from different domains:
  - Biologists
  - Chemists
  - Computer Scientists
  - Physician



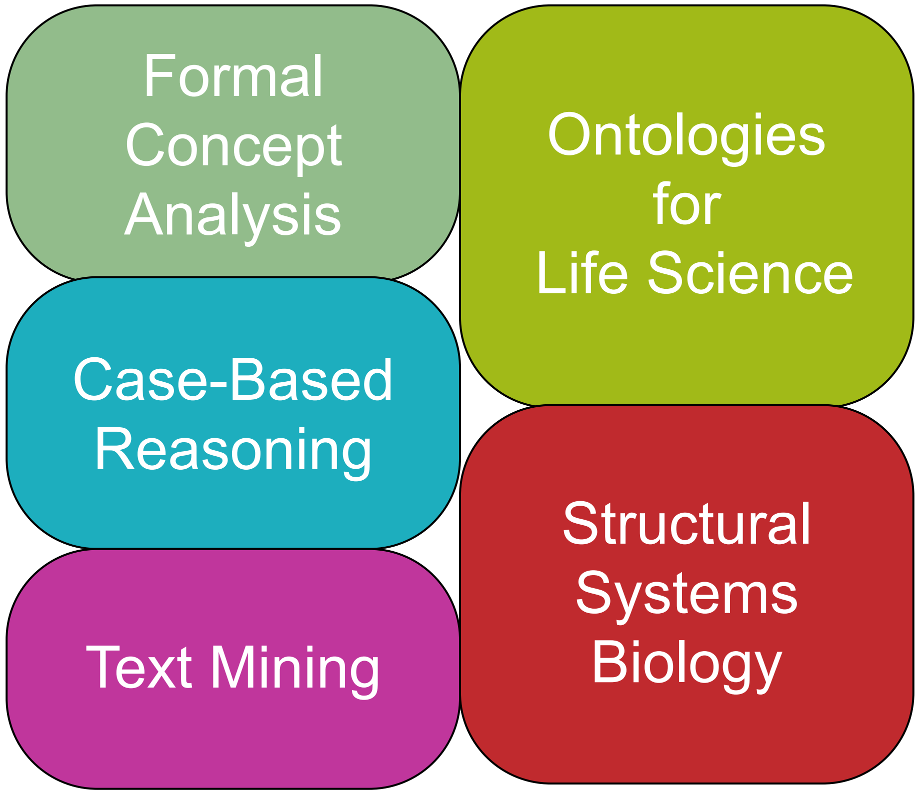
Who are these gold prospectors?

- ORPAILLEUR project-team includes a large panel of experts from different domains:
  - Biologists
  - Chemists
  - Computer Scientists
  - Physician



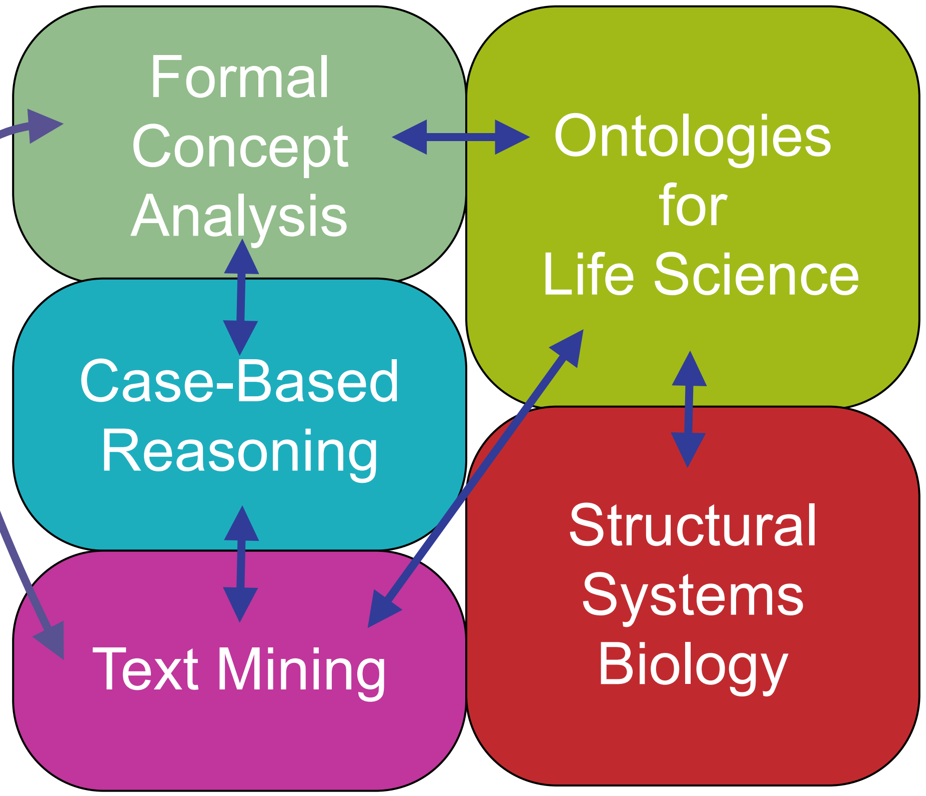
Who are these gold prospectors?

- ORPAILLEUR project-team includes a large panel of experts from different domains:
  - Biologists
  - Chemists
  - Computer Scientists
  - Physician



Who are these gold prospectors?

- ORPAILLEUR project-team includes a large panel of experts from different domains:
  - Biologists
  - Chemists
  - Computer Scientists
  - Physician



Who are these gold prospectors?

- ORPAILLEUR project-team includes a large panel of experts from different domains:
  - Biologists
  - Chemists
  - Computer Scientists
  - Physician

Knowledge Discovery
guided by
Domain Knowledge

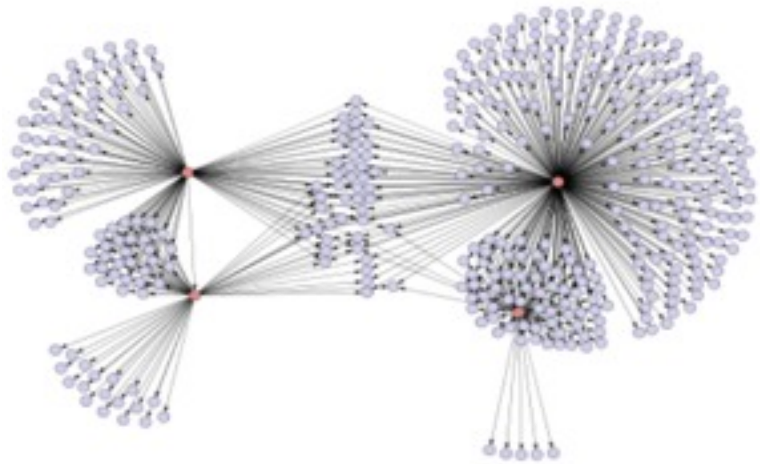
1
BIG DATA

Size does not matter, it's the way you use it!

Social Networking Websites

Biological Network: Protein Interaction

Research Collaboration Network



Product Recommendation Network via Emails

Size does not matter, it's the way you use it!

- Big data appear in all types of data we see
- But...How big is big?
  1. For statisticians: 10^3
  2. For DBers: 10^9 to 10^{15}
  3. For BigData users and data miners?
- Data may not be big, but the complexity of the data is big

  Human Genome: 3000 megabase pairs but who can really understand it?

Scalability is the key

- *"Thou shalt not Hadoop"*
  The Hadoop Grid Fallacy
- The important thing is not big data but the query you are trying to answer
- Query complexity grows faster than the number of data points
- Language semantics is the real hard problem linked to Big Data (thank you Google)

Booking a Flight Ticket

- Travel agent: What kind of flight do you prefer?
- Customer: I prefer Skyteam companies, no transits and cheap, too.
- Travel agent: Which is more important for you: the company or the price?
- Customer: The price, definitely.
- Travel agent: I have these flights.
- Customer: Wait...can I only get Airbus planes?

2
SKYLINE QUERIES

What Are Skyline Queries?

- In a database, a Skyline is a set of tuples of information (points) that are of special interest to the user
- With respect to a set of preferences!

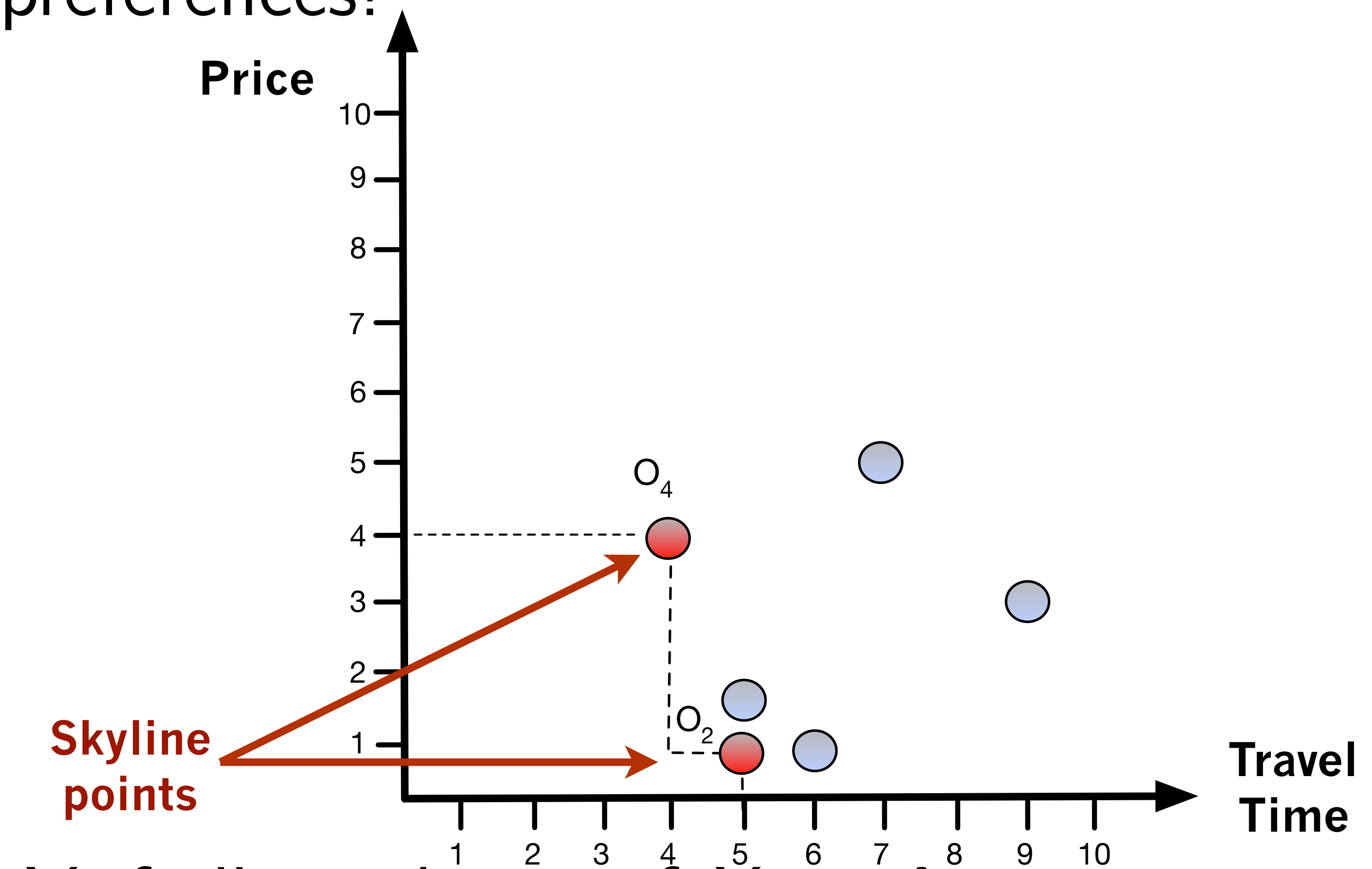
Company	**Price**	**Travel Time**
O_1	1	6
O_2	1	5
O_3	2	5
O_4	4	4
O_5	3	9
O_6	5	7

- Data point X *dominates* Y if all attributes of X are **better than or equal to** the corresponding attributes from Y
- A skyline query returns all data points that are not *dominated by others*

What Are Skyline Queries?

- In a database, a Skyline is a set of tuples of information (points) that are of special interest to the user
- With respect to a set of preferences!

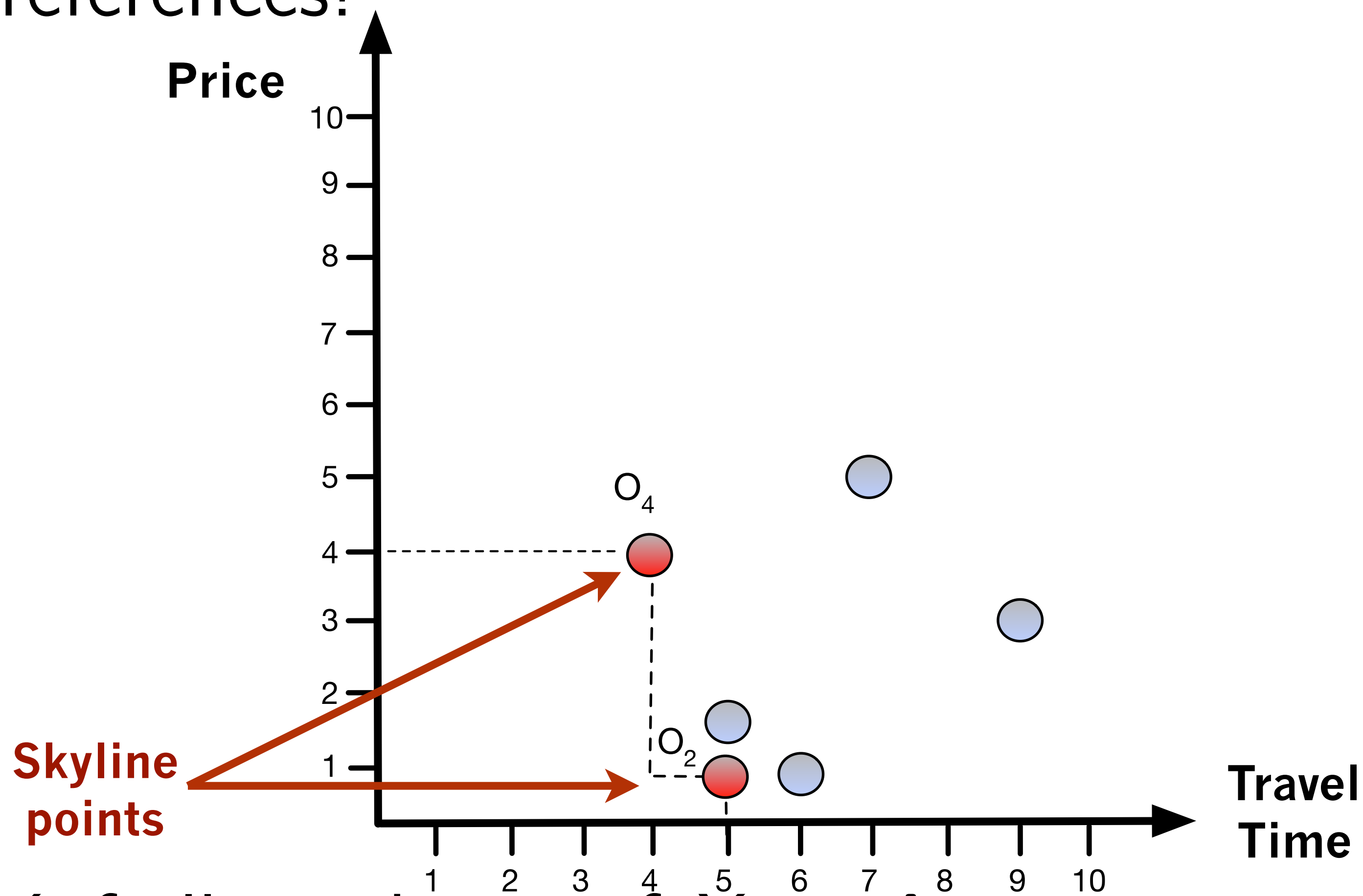
Company	Price	Travel Time
O_1	1	6
O_2	1	5
O_3	2	5
O_4	4	4
O_5	3	9
O_6	5	7



- Data point X *dominates* Y if all attributes of X are **better than or equal to** the corresponding attributes from Y
- A skyline query returns all data points that are not *dominated by others*

What Are Skyline Queries?

- In a database, a Skyline is a set of tuples of information (points) that are of special interest to the user
- With respect to a set of preferences!



- Data point X *dominates* Y if all attributes of X are **better than or equal to** the corresponding attributes from Y
- A skyline query returns all data points that are not *dominated by others*

What Are Skyline Queries?

- This is an old problem known as the maximum vector problem [KLP75, PS85], Pareto frontier
- One-dimensional Skyline = trivial because it is equivalent to computing min or max
- How to implement it ?
  1. Build it on top of relational database system. Use existing SQL Queries (poor performances)
  2. Extend SQL with a new "Skyline Operator": SKYLINE OF

What Are Skyline Queries?

```
SELECT *
FROM Hotels
WHERE city = 'Nassau'
SKYLINE OF price MIN, distance MIN;
```

Query 1

```
SELECT *
FROM Buildings
WHERE city = 'New York'
SKYLINE OF distance MIN, height MAX,
           x DIFF;
```

Query 2

```
SELECT e.name, e.salary, sum(s.volume) as volume
FROM Emp e, Sales s
WHERE e.id = s.repr AND s.year = 1999
GROUP BY e.name, e.salary
SKYLINE OF e.salary MIN, volume MAX;
```

Query 3

```
SELECT name, distance,
       (CASE WHEN price ≤ 50 THEN 'cheap'
             WHEN price > 50 THEN 'exp') AS pcat
FROM Hotels
WHERE city = 'Nassau'
SKYLINE OF pcat MIN, distance MIN;
```

Query 4

- Query1: Cheap hotels near to the beach
- Query2: High buildings close to the river (NY Skyline)
- Query3: Employees that sell a lot and have a low salary
- Query4: Cheap hotels near to the beach (only 2: one for category [cheap vs. expensive])

Skyline Exercise

QUESTION: What restaurants are in the skyline if we want the best *service*, *food*, *decor*, and of course, at the *cheapest price*?

restaurant	S	F	D	price
Summer Moon	21	25	19	47.50
Zakopane	24	20	21	56.00
Brearton Grill	15	18	20	62.00
Yamanote	22	22	17	51.50
Fenton & Pickle	16	14	10	17.50
Briar Patch BBQ	14	13	3	22.50

Skyline Exercise

QUESTION: What restaurants are in the skyline if we want the best *service*, *food*, *decor*, and of course, at the *cheapest price*?

restaurant	S	F	D	price
Summer Moon	21	25	19	47.50
Zakopane	24	20	21	56.00
Brearton Grill	15	18	20	62.00
Yamanote	22	22	17	51.50
Fenton & Pickle	16	14	10	17.50
Briar Patch BBQ	14	13	3	22.50

Skyline Exercise

QUESTION: What restaurants are in the skyline if we want the best *service*, *food*, *decor*, and of course, at the *cheapest price*?

restaurant	S	F	D	price
Summer Moon	21	25	19	47.50
Zakopane	24	20	21	56.00
Brearton Grill	15	18	20	62.00
Yamanote	22	22	17	51.50
Fenton & Pickle	16	14	10	17.50
Briar Patch BBQ	14	13	3	22.50

restaurant	S	F	D	price
Summer Moon	21	25	19	47.50
Zakopane	24	20	21	56.00
Yamanote	22	22	17	51.50
Fenton & Pickle	16	14	10	17.50

Implementation of the Skyline Queries

Nested SQL Queries

- Built on top of a relational database system
- Translate the Skyline query into a nested SQL query

```
SELECT *
FROM   Hotels h
WHERE  h.city = 'Nassau' AND NOT EXISTS(
       SELECT *
       FROM   Hotels h1
       WHERE  h1.city = 'Nassau' AND h1.distance <= h.distance AND
              h1.price <= h.price AND
              (h1.distance < h.distance OR h1.price < h.price));
```

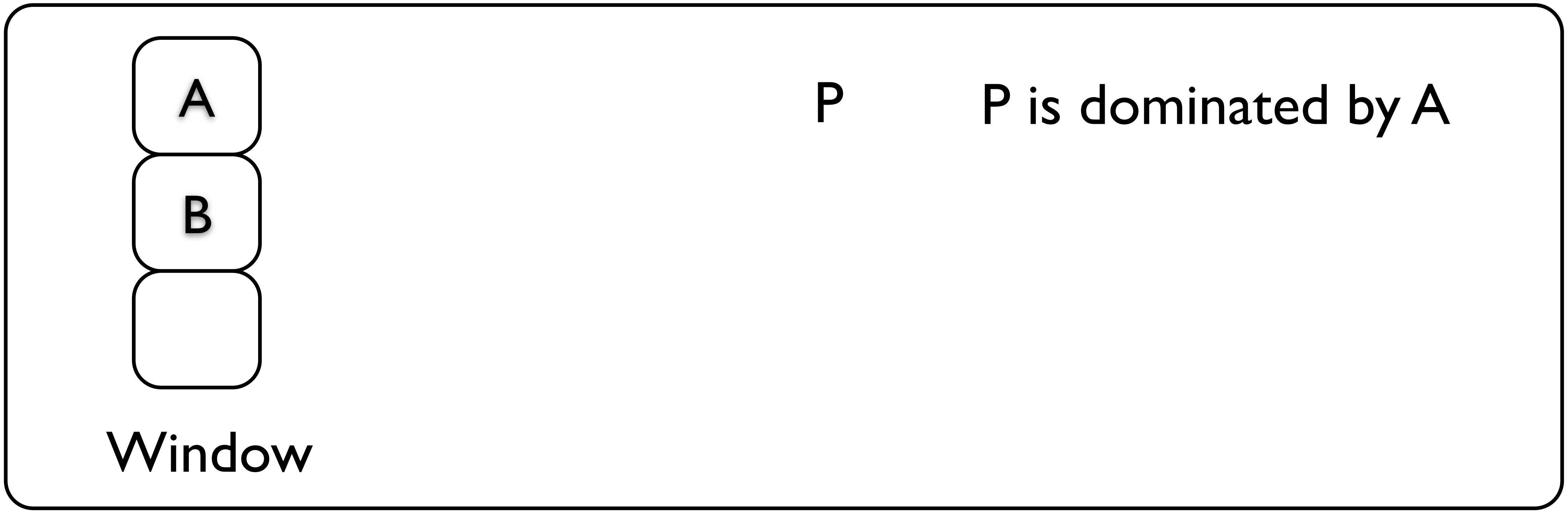
Pros: SQL what else? Cons: very poor performance

Implementation of the Skyline Queries

Block-nested-loops Algorithm (BNL) [Börzsönyi et al. 2001]

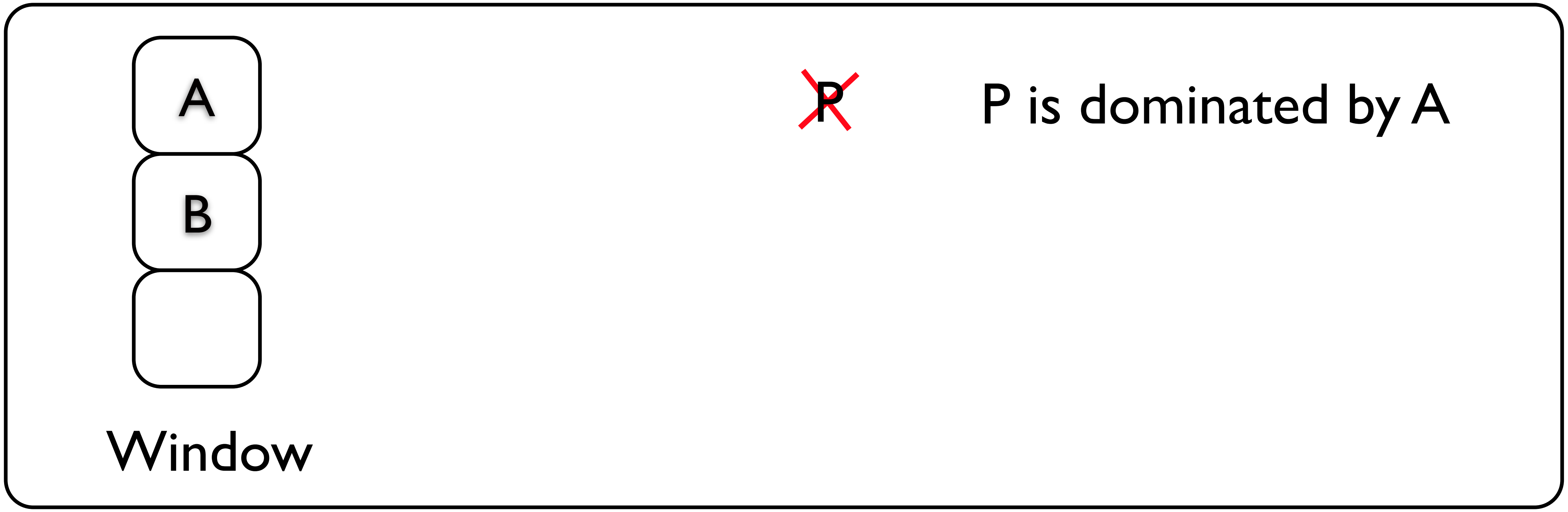
- Significantly faster than the naive approach
  Produces a block of Skyline tuples in every iteration
- Keep a window of incomparable tuples in main memory
- p is read from the input, and compared to all tuples of the window
- Based on this comparison
  1. p is dominated by a tuple within the window: p is eliminated
  2. p dominates one or more tuples in the window: tuples are eliminated and p is placed in the window
  3. p is incomparable with all the tuples in the window: p is inserted in the window if there is no space, p is written into a temporary file

Implementation of the Skyline Queries



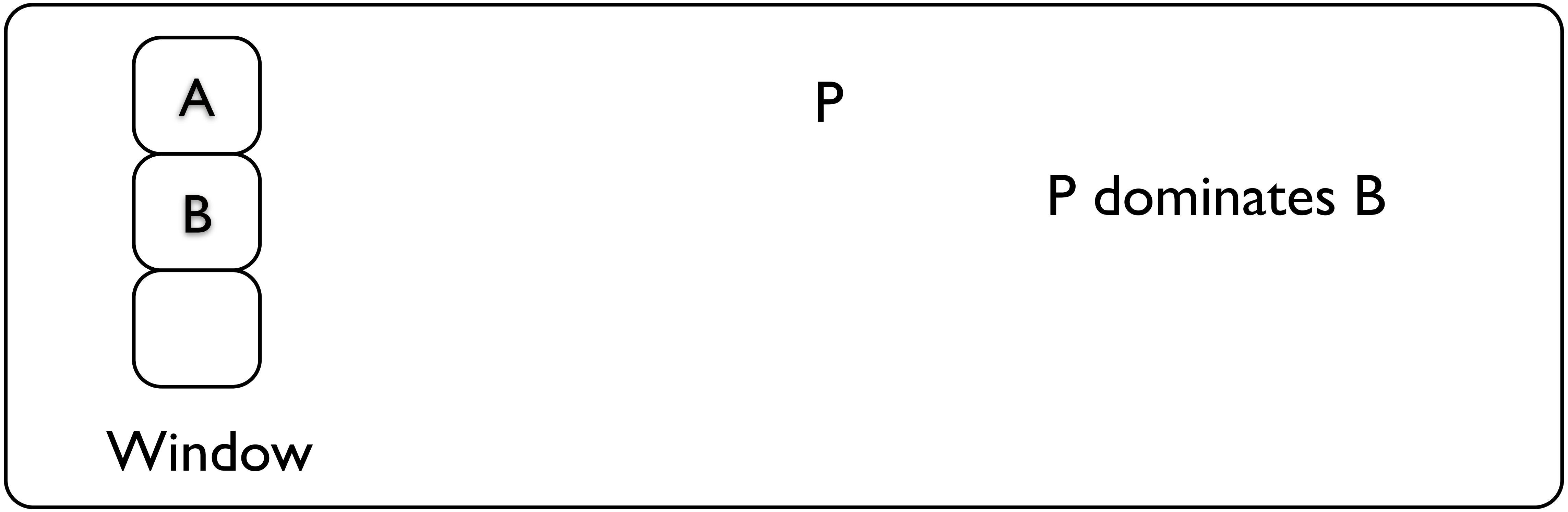
- Based on this comparison
  1. p is dominated by a tuple within the window: p is eliminated
  2. p dominates one or more tuples in the window: tuples are eliminated and p is placed in the window
  3. p is incomparable with all the tuples in the window: p is inserted in the window if there is no space, p is written into a temporary file

Implementation of the Skyline Queries



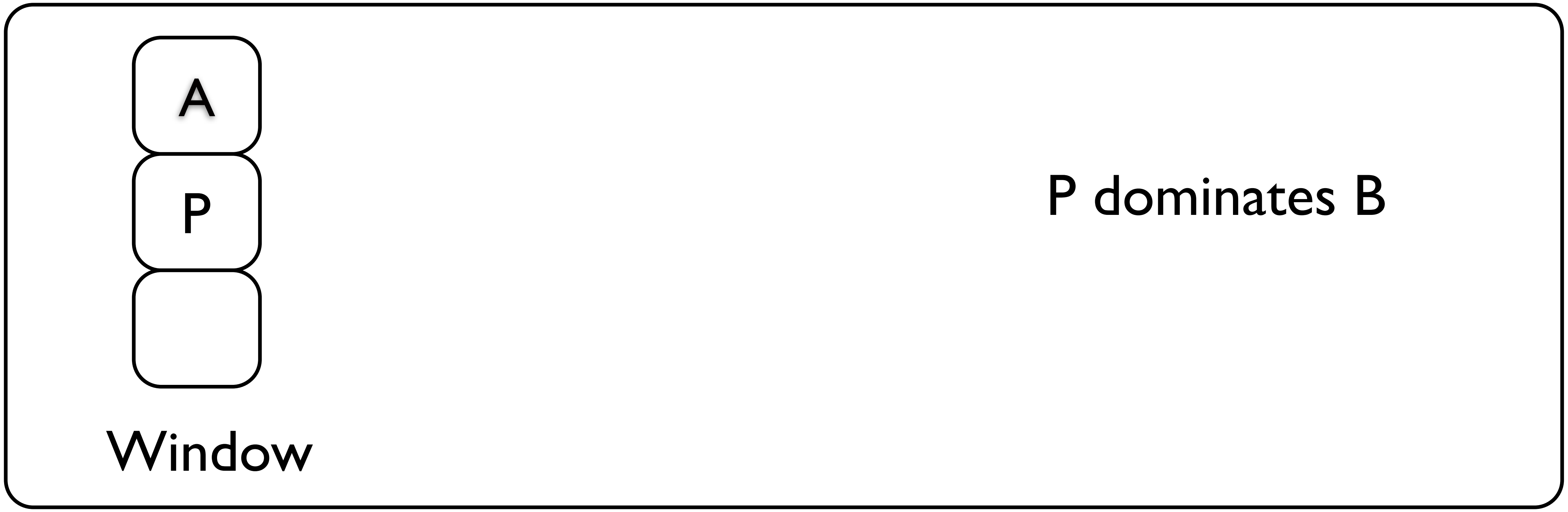
- Based on this comparison
  1. p is dominated by a tuple within the window: p is eliminated
  2. p dominates one or more tuples in the window: tuples are eliminated and p is placed in the window
  3. p is incomparable with all the tuples in the window: p is inserted in the window if there is no space, p is written into a temporary file

Implementation of the Skyline Queries



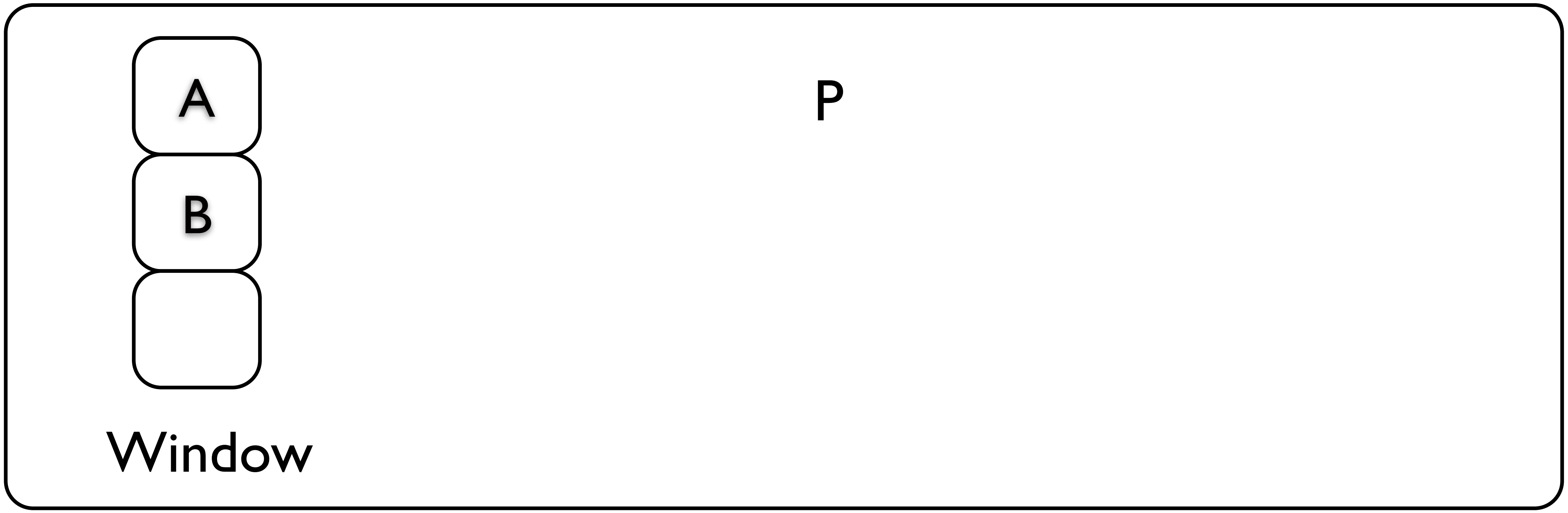
- Based on this comparison
  1. p is dominated by a tuple within the window: p is eliminated
  2. p dominates one or more tuples in the window: tuples are eliminated and p is placed in the window
  3. p is incomparable with all the tuples in the window: p is inserted in the window if there is no space, p is written into a temporary file

Implementation of the Skyline Queries



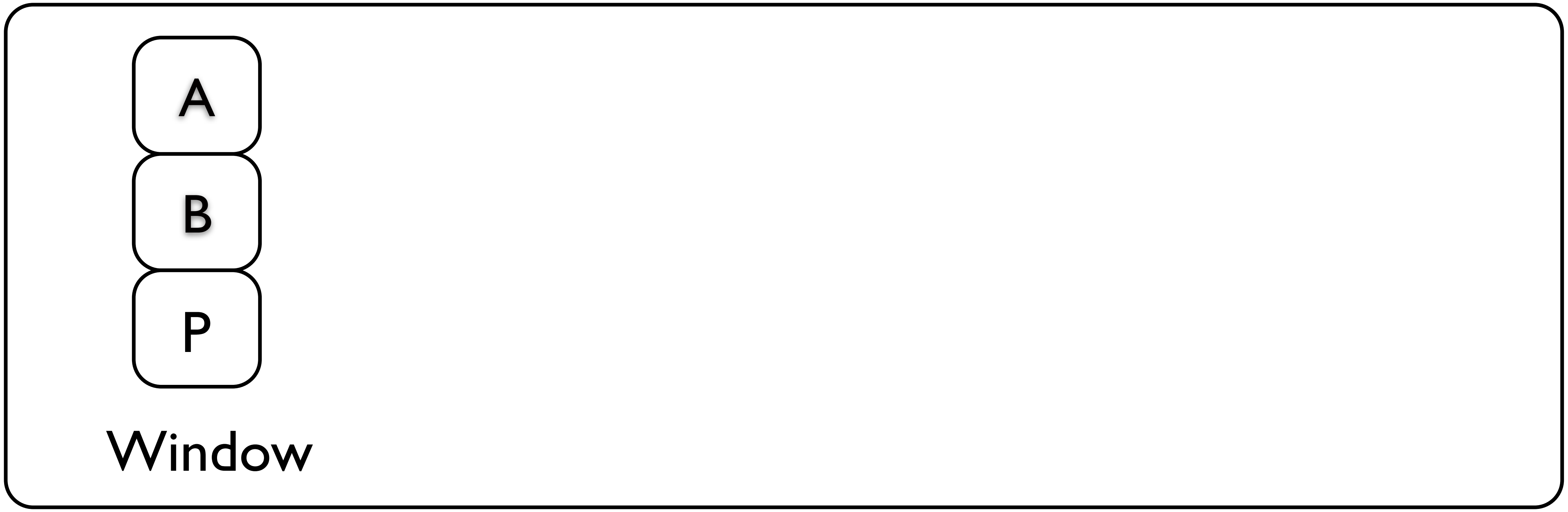
- Based on this comparison
  1. p is dominated by a tuple within the window: p is eliminated
  2. p dominates one or more tuples in the window: tuples are eliminated and p is placed in the window
  3. p is incomparable with all the tuples in the window: p is inserted in the window if there is no space, p is written into a temporary file

Implementation of the Skyline Queries



- Based on this comparison
  1. p is dominated by a tuple within the window: p is eliminated
  2. p dominates one or more tuples in the window: tuples are eliminated and p is placed in the window
  3. p is incomparable with all the tuples in the window: p is inserted in the window if there is no space, p is written into a temporary file

Implementation of the Skyline Queries



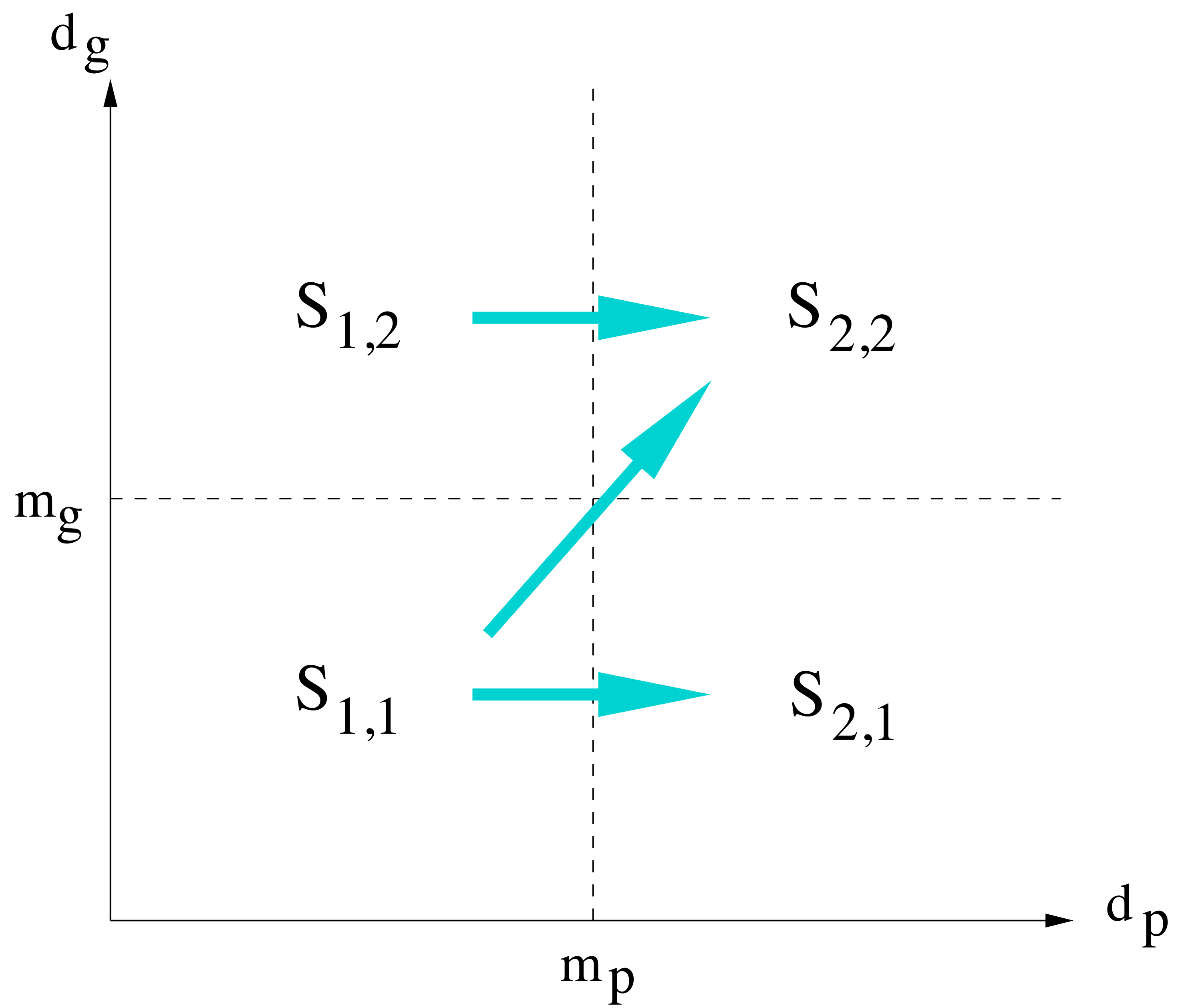
- Based on this comparison
  1. p is dominated by a tuple within the window: p is eliminated
  2. p dominates one or more tuples in the window: tuples are eliminated and p is placed in the window
  3. p is incomparable with all the tuples in the window: p is inserted in the window if there is no space, p is written into a temporary file

Implementation of the Skyline Queries

Divide and Conquer Algorithm (DC) [Börzsönyi et al. 2001]

- Theoretically the best known algorithm for the worst case
  1. Compute the median of the input for some dimension d. Divide the input into 2 partitions
  2. Compute the Skylines S1 and S2 of P1 and P2 Recursively apply the whole algorithm to P1 and P2
  3. Merge S1 and S2 to compute the overall Skyline. Eliminate all the tuples of S2 which are dominated by tuples of S1 (No tuples of S1 can be dominated by tuples of S2 as tuples in S1 have a better d1 value)

Implementation of the Skyline Queries



References

- *On finding the maxima of a set of vectors*, H. T. Kung, F. Luccio, and F. P. Preparata. Journal of the ACM, 22(4):469-476, 1975
- *The Skyline Operator*, S. Börzsönyi, D. Kossmann, K. Stocker [ICDE 2001]
- *Skyline Queries and it variations*, Jagan Sankaranarayanan, [CMSC828S]
- *An Optimal and Progressive Algorithm for Skyline Queries*, D.Papadias, Y.Tao, G.Fu, B. Seeger, [SIGMOD 2003]

3
SKYCUBES

Subspaces Skylines

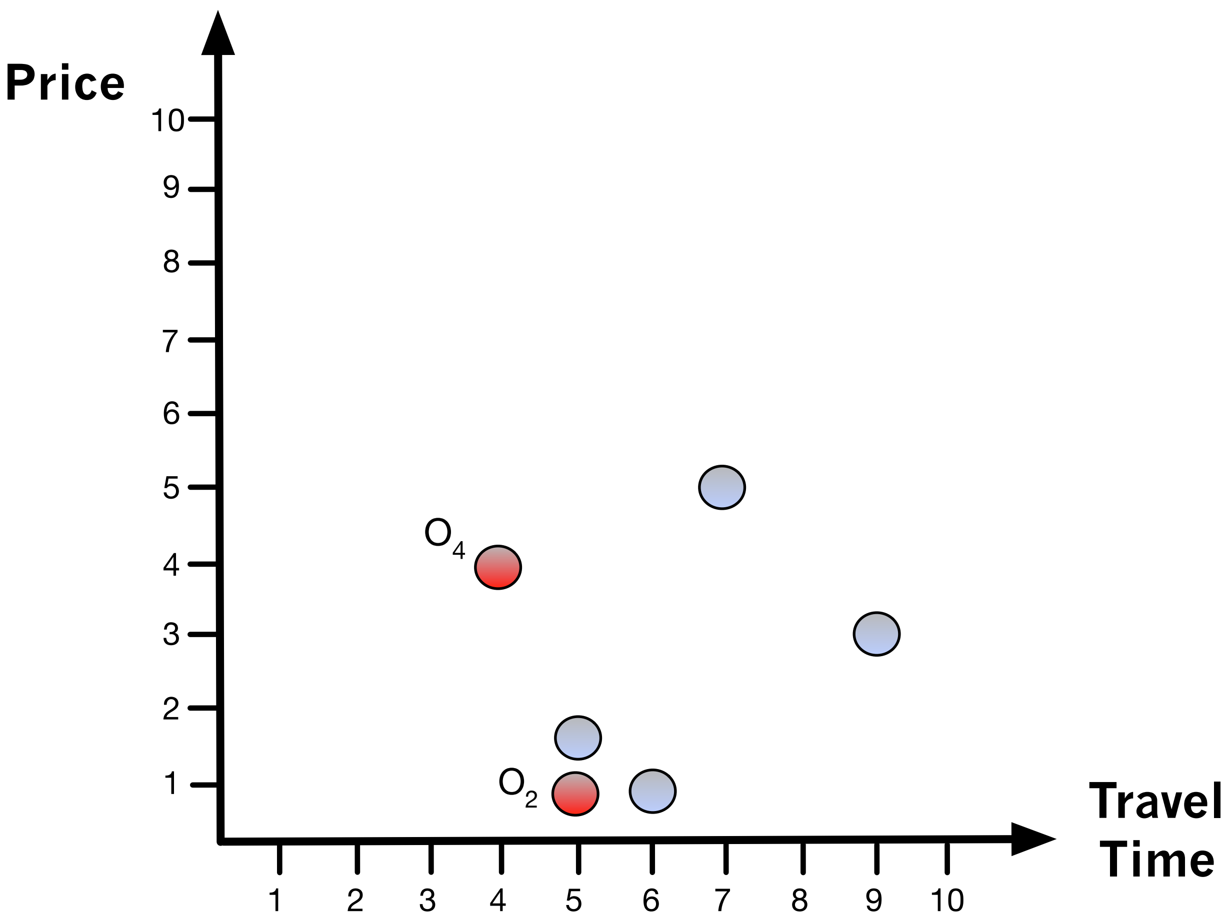
- What if a user is not interested by the full-space skyline but **only by some precise subsets**?

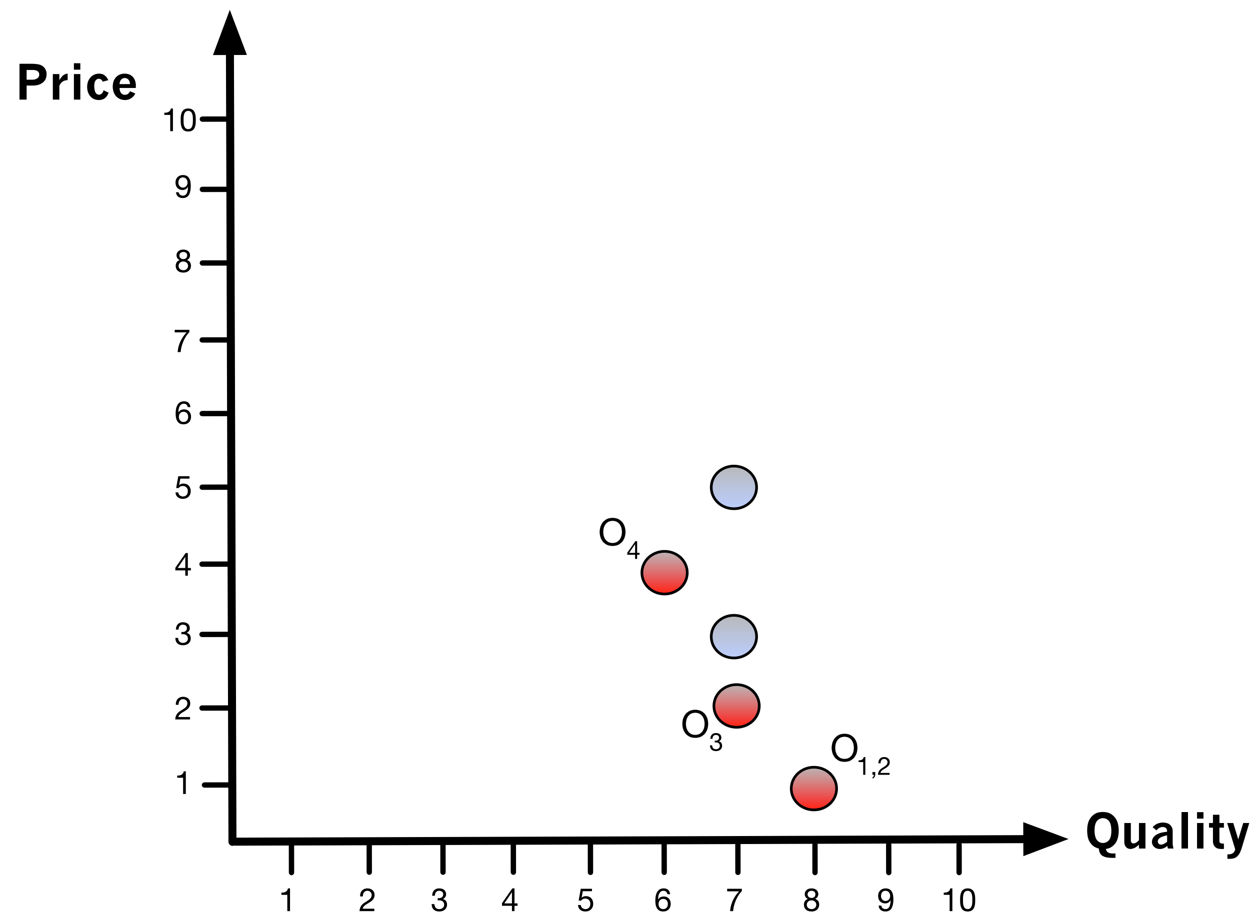
Company	Price	Transits	Travel Time	Quality
O_1	1	3	6	8
O_2	1	3	5	8
O_3	2	4	5	7
O_4	4	4	4	6
O_5	3	9	9	7
O_6	5	8	7	7

- Results may be completely **different**
- For online systems, **on-the-fly** computation is **not efficient**

Subspaces Skylines

- What if a user is not interested by the full-space skyline but **only by some precise subsets**?

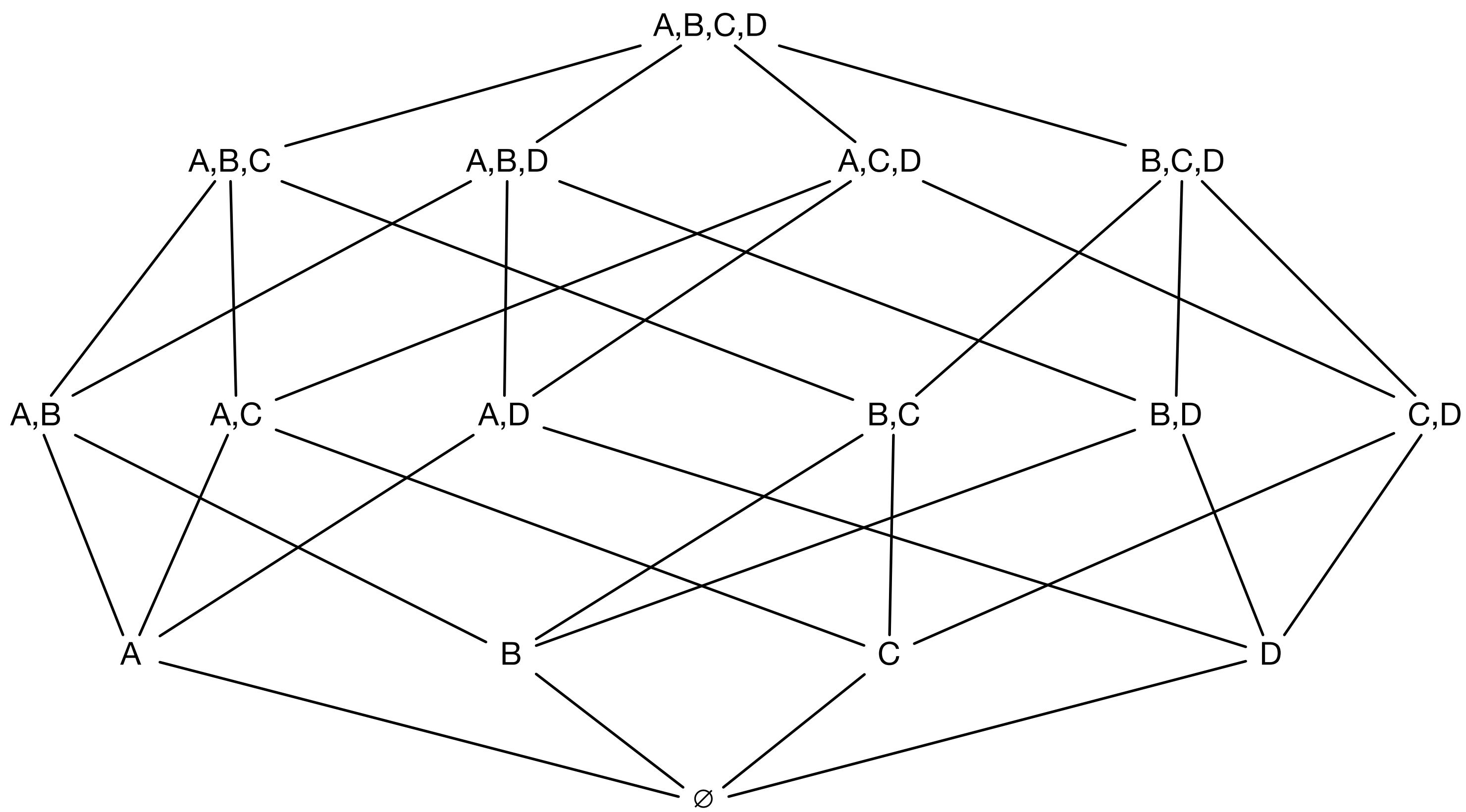




- Results may be completely **different**
- For online systems, **on-the-fly** computation is **not efficient**

The Skycube

- Introduced by Yuan et al. in VLDB 2005
- **Goal**: Compute and store all subspace skyline results
  - For d dimensions, the collection will contain $2^d - 1$ subspaces

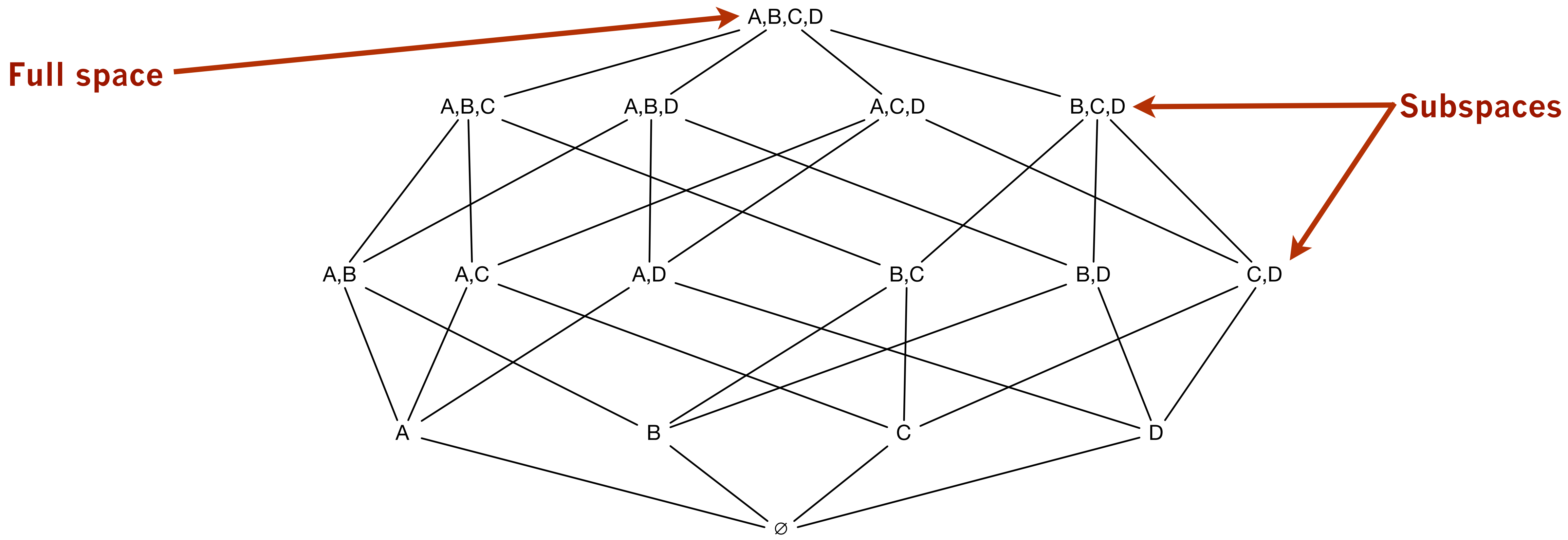


Applications

- Hotel/ Air ticket recommendations
- Any product that allow users preferences among attributes

The Skycube

- Introduced by Yuan et al. in VLDB 2005
- **Goal**: Compute and store all subspace skyline results
  - For d dimensions, the collection will contain $2^d - 1$ subspaces



Applications

- Hotel/ Air ticket recommendations
- Any product that allow users preferences among attributes

The Skycube

How to compute a skycube efficiently?

Share results between the different subspaces skylines computations

- BUS algorithm (bottom-up)
- TDS algorithm (top-down)

- Skyey algorithm

Succinct summarization subspace skylines based on semantics

- Stellar algorithm

Motivation

Observations

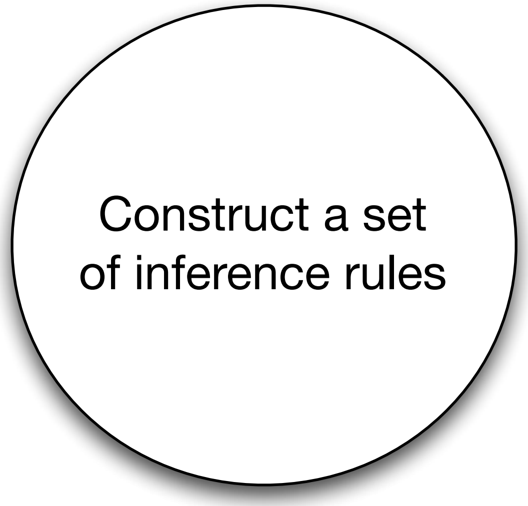
- Domination tests = major cost in skyline computation
- Fundamental theoretical questions
  - Is it possible to *derive* skylines without domination tests?
  - Is it possible to *avoid the computation* of a group of subspaces?

Challenges

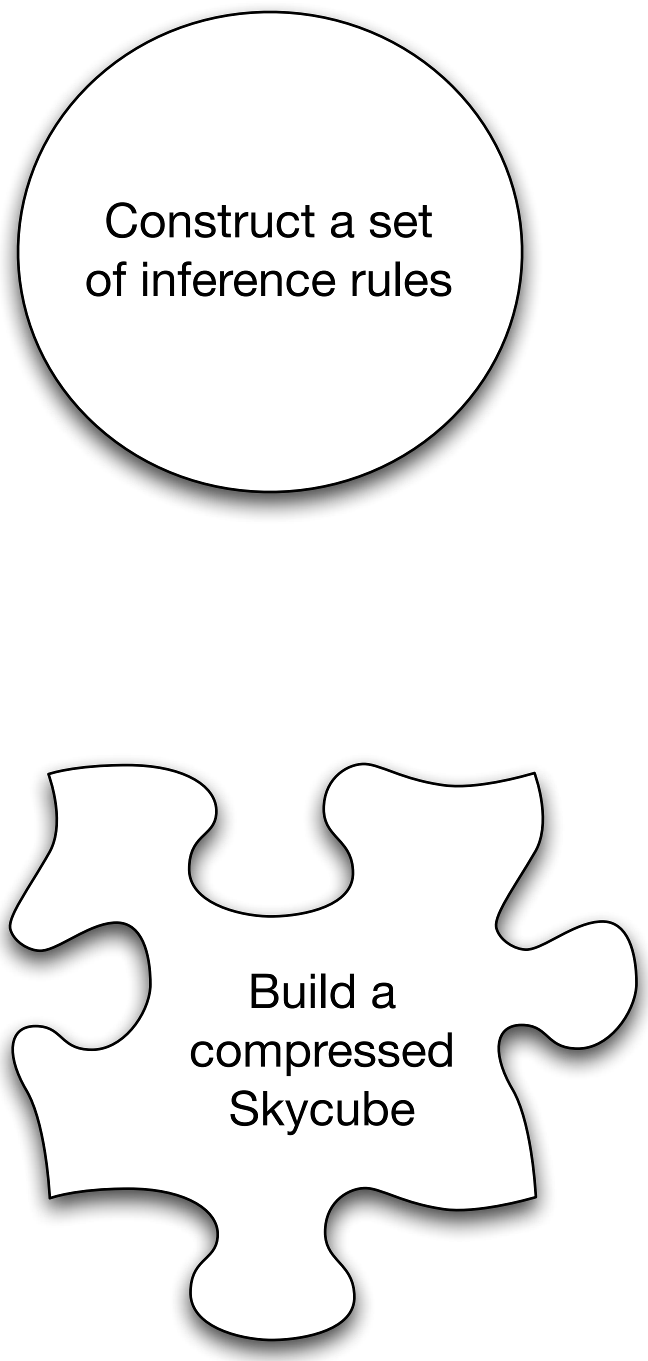
1. **Minimize** domination tests
2. **Compress** skycube
   - limit subspaces computations **to the maximum**
3. Compute skycubes for **high-dimensional** data sets

Orion Framework

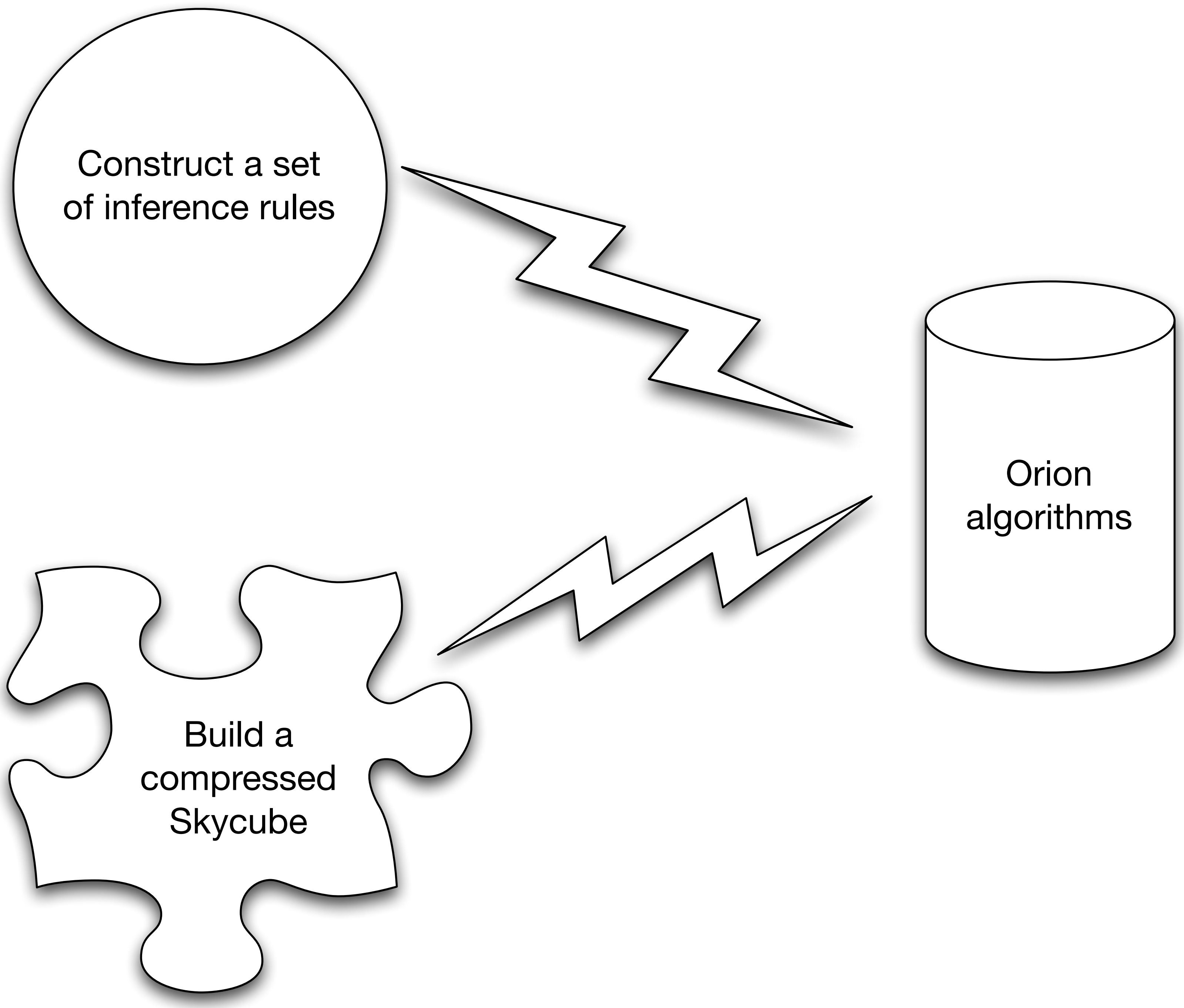
Orion Framework



Orion Framework



Orion Framework



Preliminary Definitions

Definition (Indistinct and incomparable skyline)

- $p \in SKY(\mathcal{U})$ is an **indistinct skyline** in $\mathcal{U}$ if $\exists q \in SKY(\mathcal{U})$ such that $p \neq q$ and $p =_{\mathcal{U}} q$.
- $p \in SKY(\mathcal{U})$ is an **incomparable skyline** in $\mathcal{U}$ if p is incomparable to any other points in $SKY(\mathcal{U})$.

Definition (Types)

$\mathcal{U}$ is a **type I subspace** if all points in $SKY(\mathcal{U})$ are indistinct from each other

Preliminary Definitions

Company	Price
O_1	1
O_2	1

Definition (Indistinct and incomparable skyline)

- $p \in SKY(\mathcal{U})$ is an **indistinct skyline** in $\mathcal{U}$ if $\exists q \in SKY(\mathcal{U})$ such that $p \neq q$ and $p =_{\mathcal{U}} q$.
- $p \in SKY(\mathcal{U})$ is an **incomparable skyline** in $\mathcal{U}$ if p is incomparable to any other points in $SKY(\mathcal{U})$.

Definition (Types)

$\mathcal{U}$ is a **type I subspace** if all points in $SKY(\mathcal{U})$ are indistinct from each other

Preliminary Definitions

Definition (Indistinct and incomparable skyline)

- $p \in SKY(\mathcal{U})$ is an **indistinct skyline** in $\mathcal{U}$ if $\exists q \in SKY(\mathcal{U})$ such that $p \neq q$ and $p =_{\mathcal{U}} q$.
- $p \in SKY(\mathcal{U})$ is an **incomparable skyline** in $\mathcal{U}$ if p is incomparable to any other points in $SKY(\mathcal{U})$.

Company	**Price**
O_1	1
O_2	1

Company	**Price**	**Travel Time**	**Quality**
O_2	1	5	8
O_3	2	5	7
O_4	4	4	6

Definition (Types)

$\mathcal{U}$ is a **type I subspace** if all points in $SKY(\mathcal{U})$ are indistinct from each other

Preliminary Definitions

Definition (Indistinct and incomparable skyline)

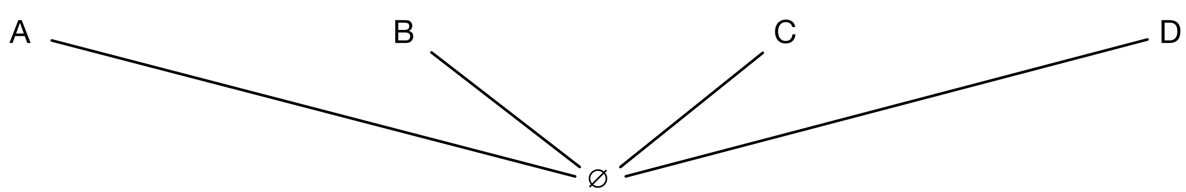
- $p \in SKY(\mathcal{U})$ is an **indistinct skyline** in $\mathcal{U}$ if $\exists q \in SKY(\mathcal{U})$ such that $p \neq q$ and $p =_{\mathcal{U}} q$.
- $p \in SKY(\mathcal{U})$ is an **incomparable skyline** in $\mathcal{U}$ if p is incomparable to any other points in $SKY(\mathcal{U})$.

Company	Price
O_1	1
O_2	1

Company	Price	Travel Time	Quality
O_2	1	5	8
O_3	2	5	7
O_4	4	4	6

Definition (Types)

$\mathcal{U}$ is a **type I subspace** if all points in $SKY(\mathcal{U})$ are indistinct from each other



Derivation Rules

Observation

Skyline membership monotonicity does not hold **in general**

- If p belongs to $SKY(\mathcal{U})$ and $SKY(\mathcal{W})$ such that $\mathcal{U} \subset \mathcal{W}$
- p may not belong to $SKY(\mathcal{V})$ where $\mathcal{U} \subset \mathcal{V} \subset \mathcal{W}$

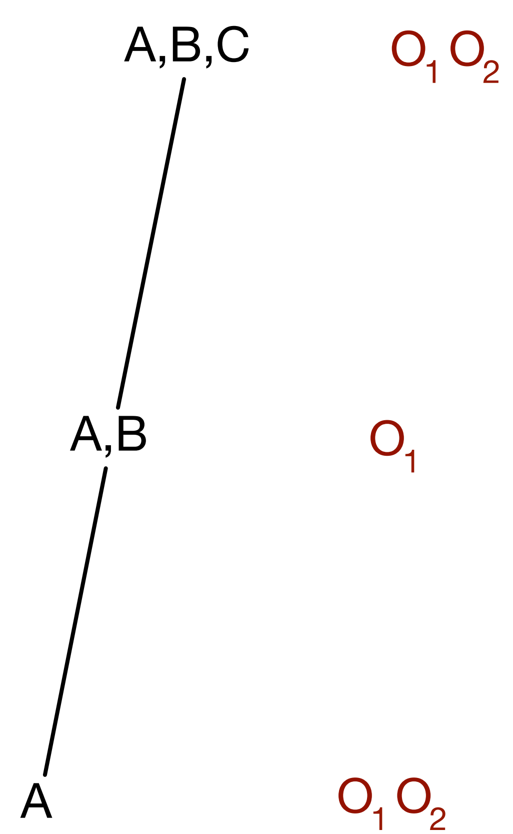
- Use a special case for efficient derivation rules

Derivation Rules

Observation

Skyline membership monotonicity does not hold **in general**

- If p belongs to $SKY(\mathcal{U})$ and $SKY(\mathcal{W})$ such that $\mathcal{U} \subset \mathcal{W}$
- p may not belong to $SKY(\mathcal{V})$ where $\mathcal{U} \subset \mathcal{V} \subset \mathcal{W}$



- Use a special case for efficient derivation rules

Derivation Rules

Theorem (Type I derivation rule)

For $\mathcal{U}$ and $\mathcal{V}$ such that $SKY(\mathcal{U}) \cap SKY(\mathcal{V}) \neq \emptyset$, if $\mathcal{U}$ and $\mathcal{V}$ are type I subspaces, then $\mathcal{U} \cup \mathcal{V}$ is a type I subspace, and

$$SKY(\mathcal{U}) \cap SKY(\mathcal{V}) = SKY(\mathcal{U} \cup \mathcal{V})$$

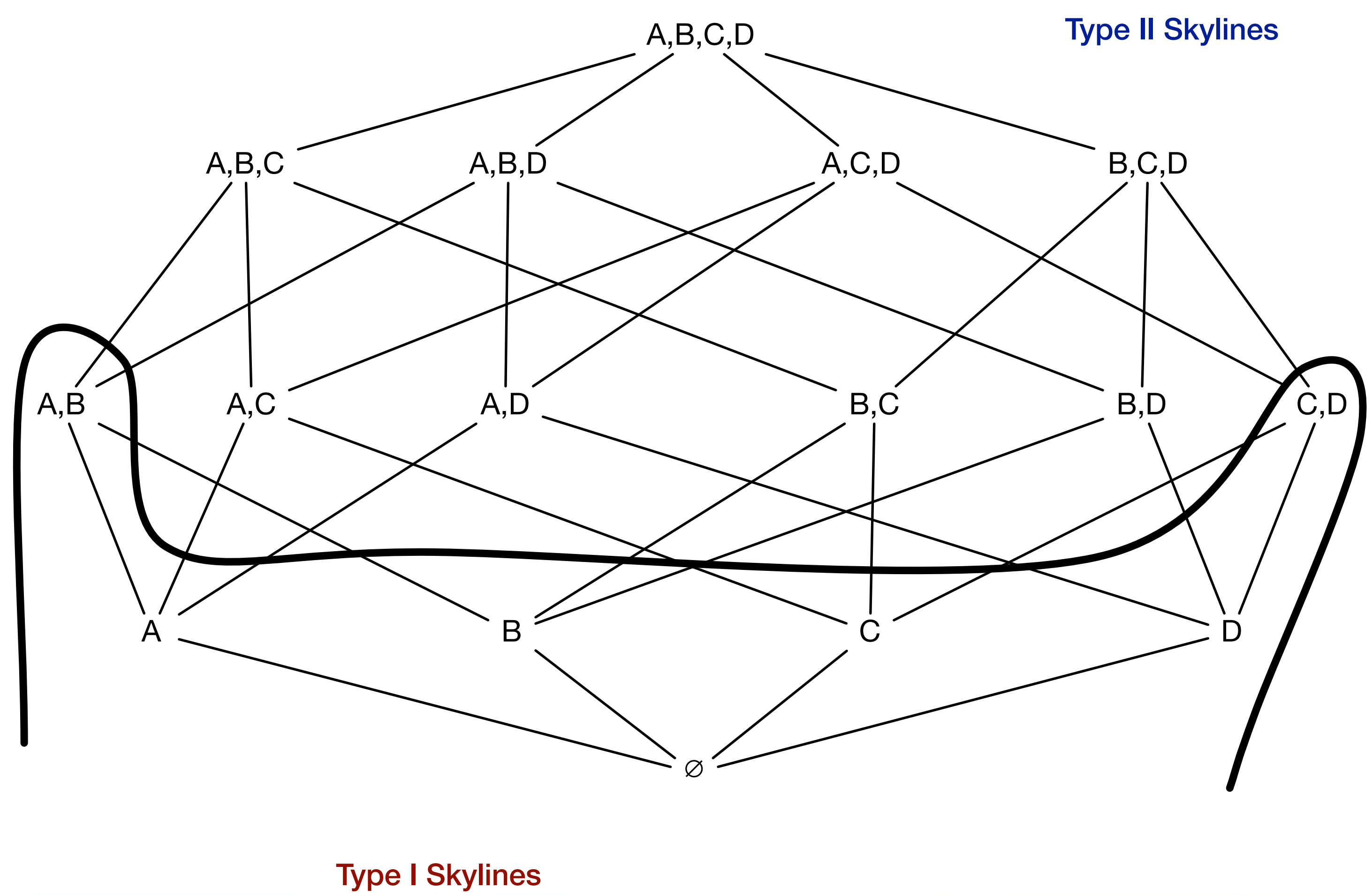
Derivation Rules

Theorem (Type I derivation rule)

For $\mathcal{U}$ and $\mathcal{V}$ such that $SKY(\mathcal{U}) \cap SKY(\mathcal{V}) \neq \emptyset$, if $\mathcal{U}$ and $\mathcal{V}$ are type I subspaces, then $\mathcal{U} \cup \mathcal{V}$ is a type I subspace, and

$$SKY(\mathcal{U}) \cap SKY(\mathcal{V}) = SKY(\mathcal{U} \cup \mathcal{V})$$

	A	B	C	D
Company	**Price**	**Transits**	**Travel Time**	**Quality**
O_1	1	3	6	8
O_2	1	3	5	8
O_3	2	4	5	7
O_4	4	4	4	6
O_5	3	9	9	7
O_6	5	8	7	7



Derivation Rules

Theorem (Incomparability rule)

If p is an incomparable skyline in subspace $\mathcal{U}$, then for any subspace $\mathcal{V}$ such that $\mathcal{U} \subseteq \mathcal{V}$

$$p \in SKY(\mathcal{V})$$

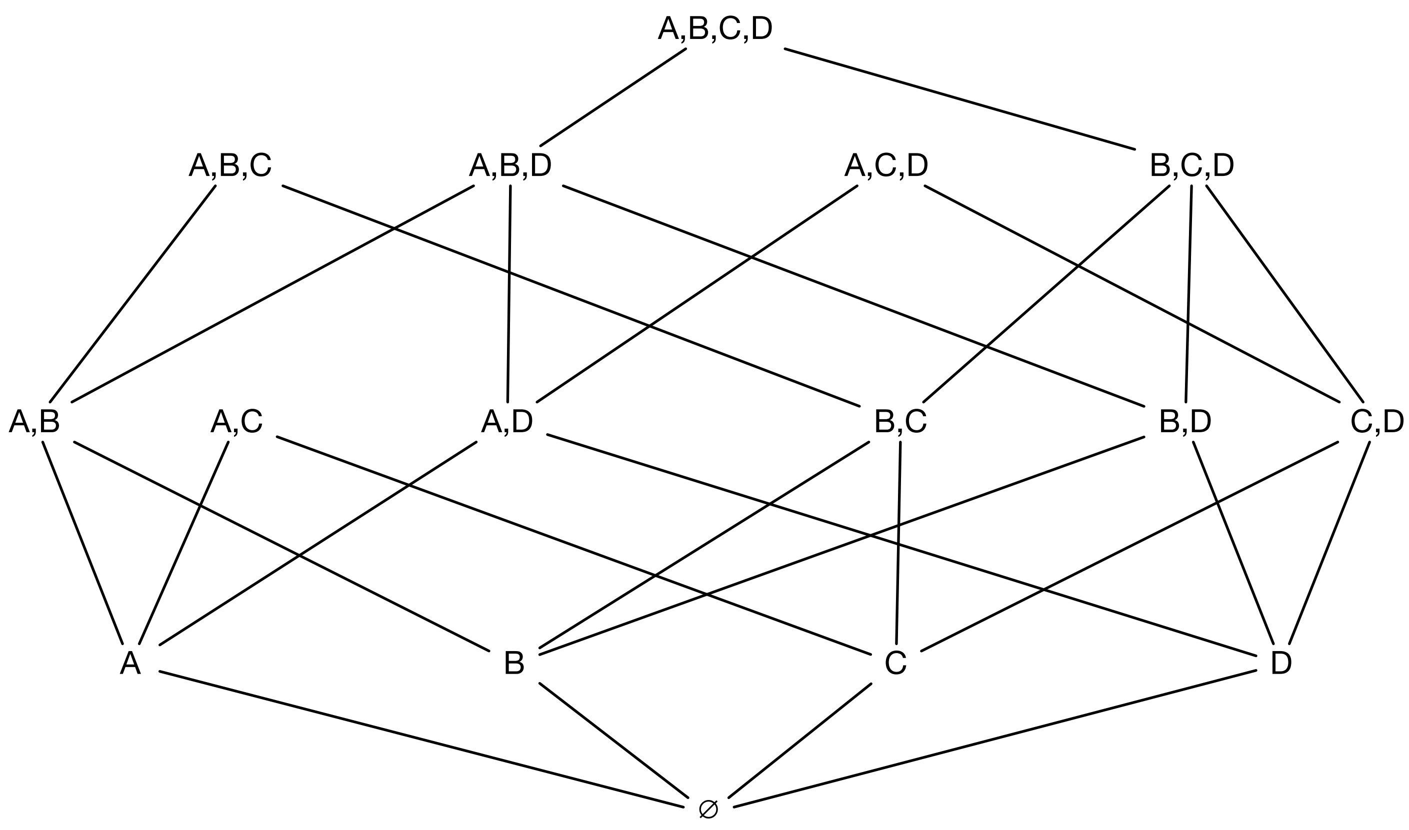
Derivation Rules

Theorem (Incomparability rule)

If p is an incomparable skyline in subspace $\mathcal{U}$, then for any subspace $\mathcal{V}$ such that $\mathcal{U} \subseteq \mathcal{V}$

$$p \in SKY(\mathcal{V})$$

	A	B	C	D
Company	**Price**	**Transits**	**Travel Time**	**Quality**
O_1	1	3	6	8
O_2	1	3	5	8
O_3	2	4	5	7
O_4	4	4	4	6
O_5	3	9	9	7
O_6	5	8	7	7



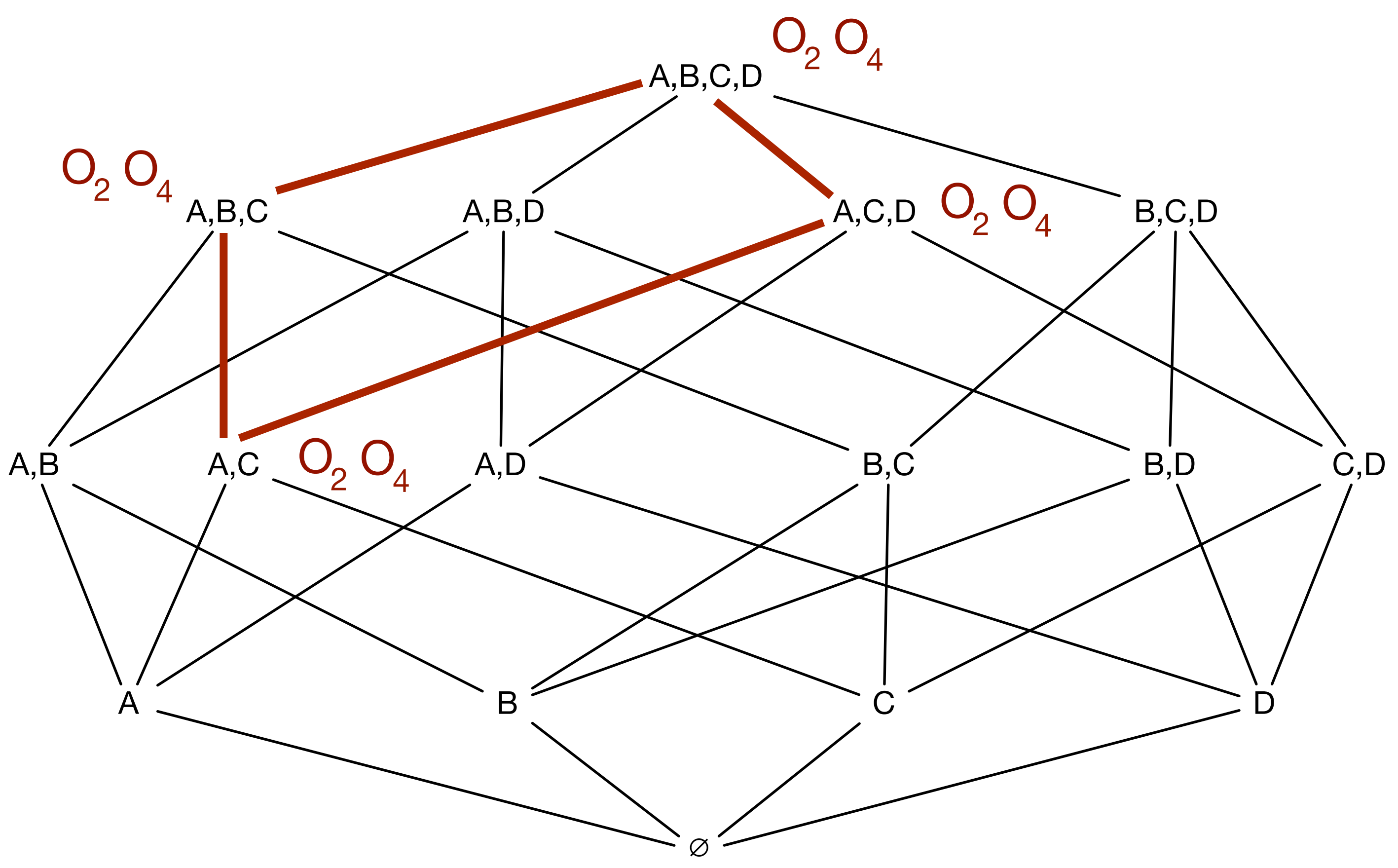
Derivation Rules

Theorem (Incomparability rule)

If p is an incomparable skyline in subspace $\mathcal{U}$, then for any subspace $\mathcal{V}$ such that $\mathcal{U} \subseteq \mathcal{V}$

$$p \in SKY(\mathcal{V})$$

	A	B	C	D
Company	**Price**	**Transits**	**Travel Time**	**Quality**
O_1	1	3	6	8
O_2	1	3	5	8
O_3	2	4	5	7
O_4	4	4	4	6
O_5	3	9	9	7
O_6	5	8	7	7



Improved Domination Tests

Optimization

With the two presented rules, **only a small number of skylines** need to be computed using domination tests

Minimize the number of candidates for domination tests

Definition (Entailed candidate)

A point p is an **entailed candidate** in a subspace $\mathcal{U}$ if $\forall d_i \in \mathcal{U}$, $min_{SKY(\mathcal{U})}(d_i) \leq p(d_i) \leq max_{SKY(\mathcal{U})}(d_i)$

Improved Domination Tests

Optimization

With the two presented rules, **only a small number of skylines** need to be computed using domination tests

Minimize the number of candidates for domination tests

Definition (Entailed candidate)

A point p is an **entailed candidate** in a subspace $\mathcal{U}$ if $\forall d_i \in \mathcal{U}$, $min_{SKY(\mathcal{U})}(d_i) \leq p(d_i) \leq max_{SKY(\mathcal{U})}(d_i)$

	A	B	C	D
Company	**Price**	**Transits**	**Travel Time**	**Quality**
O_1	1	3	6	8
O_2	1	3	5	8
O_3	2	4	5	7
O_4	4	4	4	6
O_5	3	9	9	7
O_6	5	8	7	7

Improved Domination Tests

Optimization

With the two presented rules, **only a small number of skylines** need to be computed using domination tests

Minimize the number of candidates for domination tests

Definition (Entailed candidate)

A point p is an **entailed candidate** in a subspace $\mathcal{U}$ if $\forall d_i \in \mathcal{U}$, $min_{SKY(\mathcal{U})}(d_i) \leq p(d_i) \leq max_{SKY(\mathcal{U})}(d_i)$

	A	B	C	D
Company	**Price**	**Transits**	**Travel Time**	**Quality**
O_1	1	3	6	8
O_2	1	3	5	8
O_3	2	4	5	7
O_4	4	4	4	6
O_5	3	9	9	7
O_6	5	8	7	7

SKY(AD) = O_1 O_2 O_4

Next candidates ?

All points such that:

$1 \leq p(A) \leq 4$

and

$6 \leq p(D) \leq 8$

Orion Algorithm

- Compute the skycube in a bottom-up manner and level-wise
- Apply the derivation rules then apply domination tests if needed

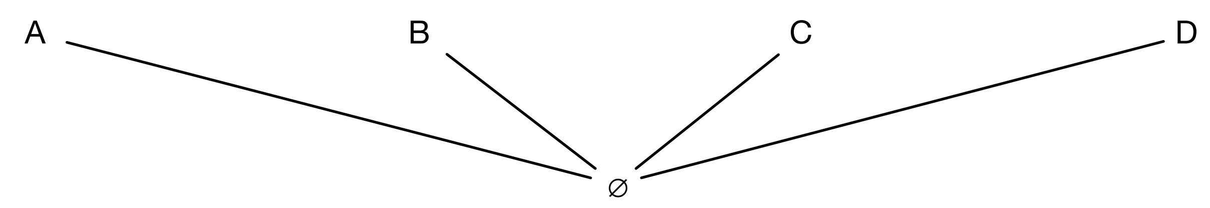
Orion Algorithm

- Compute the skycube in a bottom-up manner and level-wise
- Apply the derivation rules then apply domination tests if needed

$\varnothing$

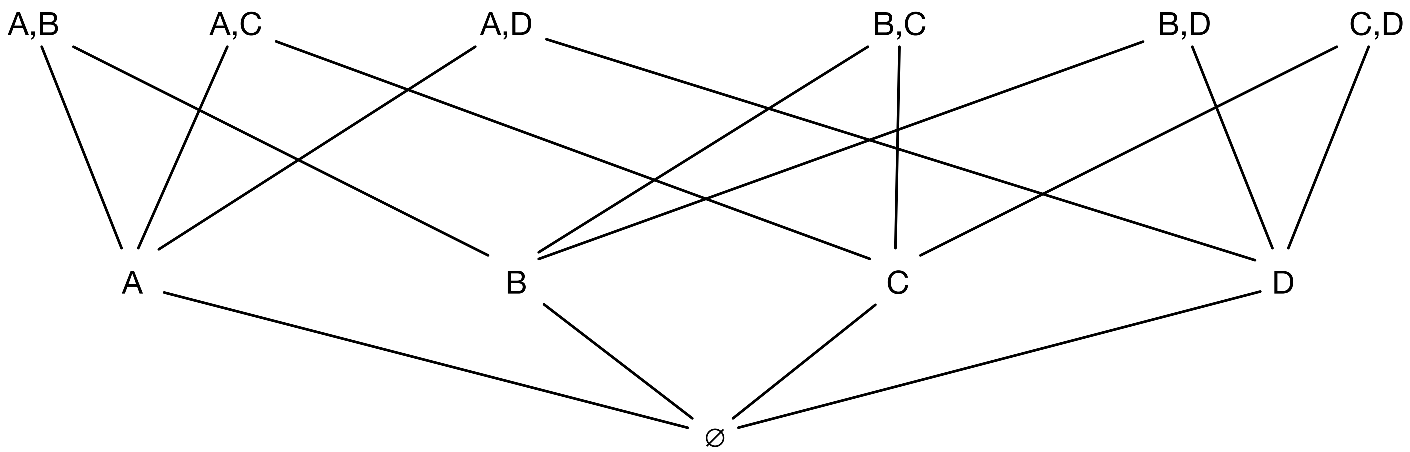
Orion Algorithm

- Compute the skycube in a bottom-up manner and level-wise
- Apply the derivation rules then apply domination tests if needed



Orion Algorithm

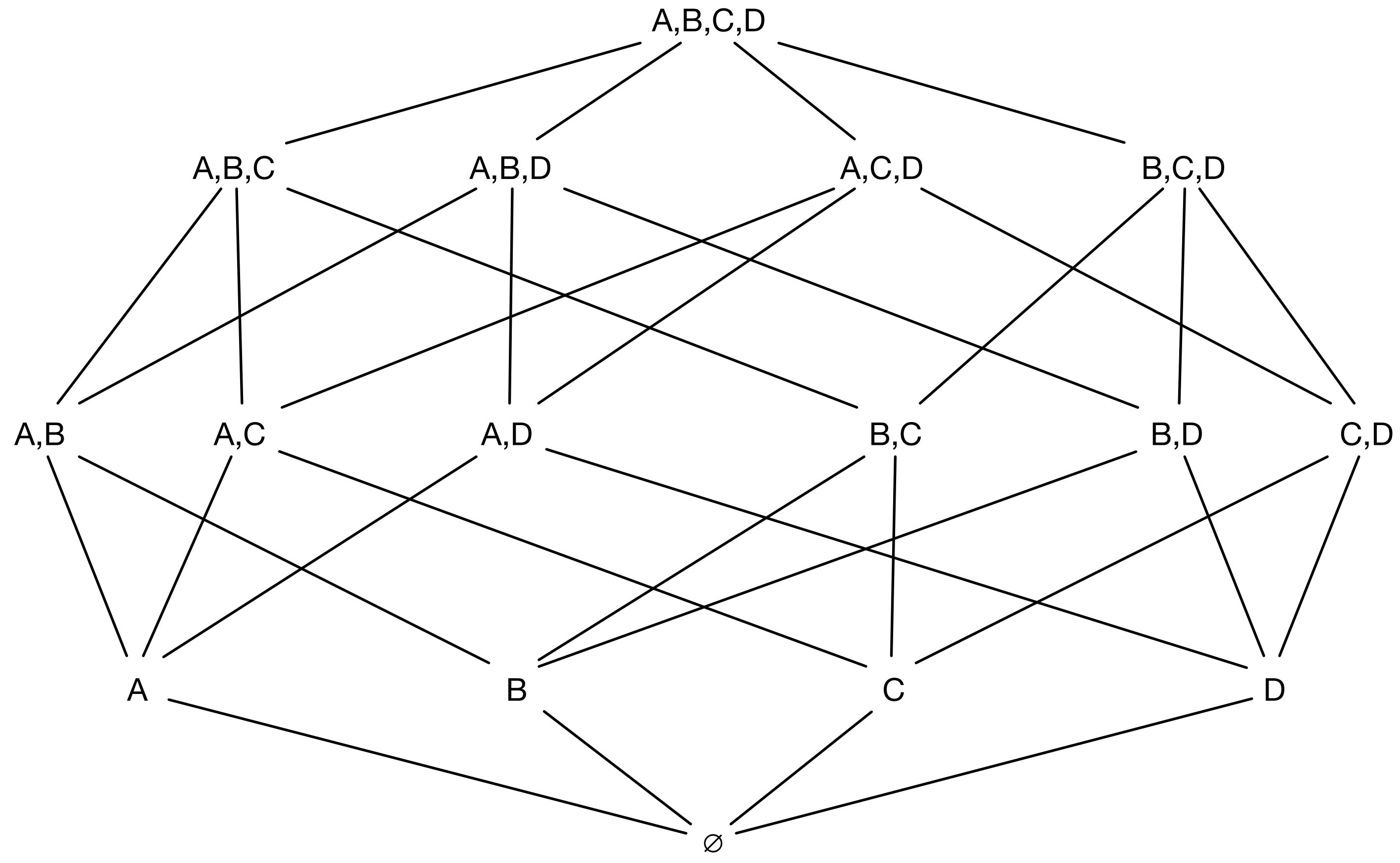
- Compute the skycube in a bottom-up manner and level-wise
- Apply the derivation rules then apply domination tests if needed



Bitmap intersections for type I rule
Incomparable skylines are propagated step-by-step

Orion Algorithm

- Compute the skycube in a bottom-up manner and level-wise
- Apply the derivation rules then apply domination tests if needed



Bitmap intersections for type I rule
Incomparable skylines are propagated step-by-step

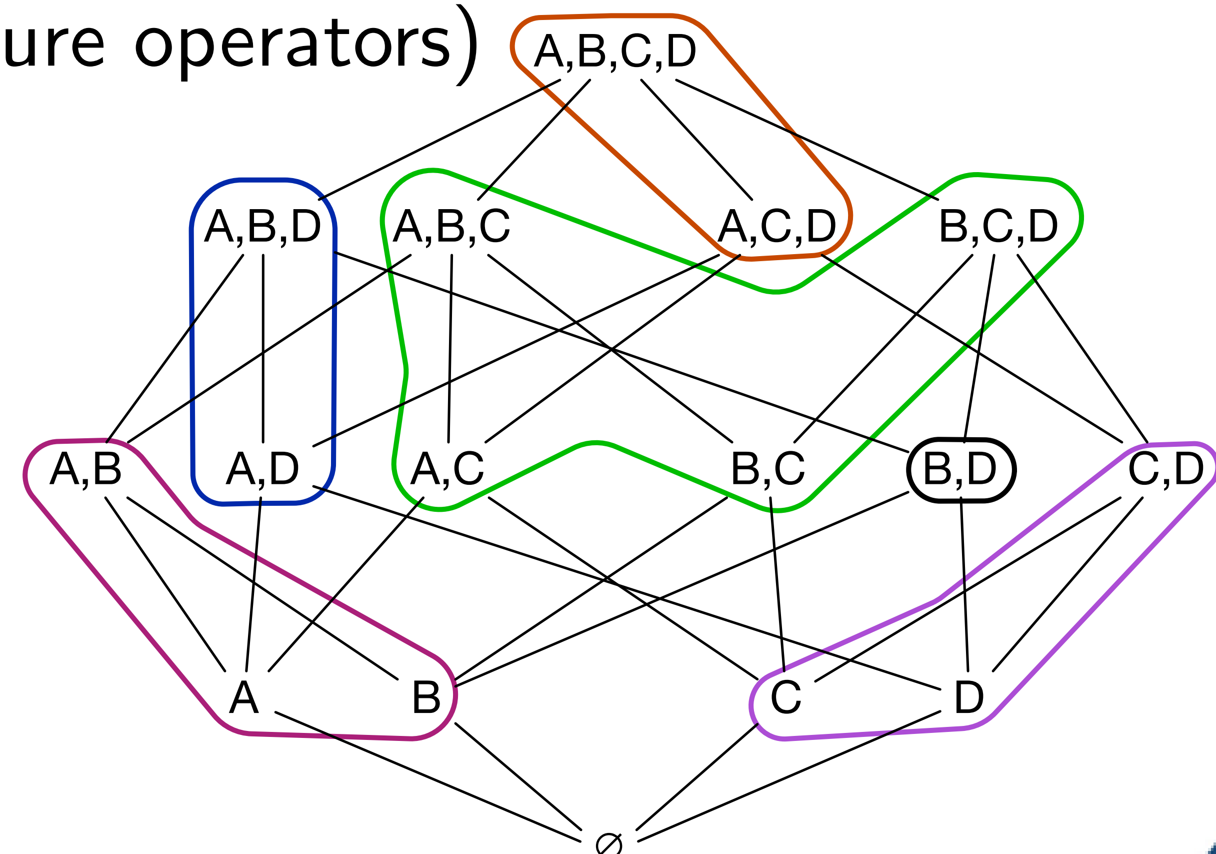
Compressed Skycube

- Deriving skylines is a good idea but...
- ...our second **goal** is to **avoid computing all** subspaces

Basic Idea

- Reduce the number of processed subspaces using notions from **Formal Concept Analysis** (closure operators)

Company	Price	Transits	Travel Time	Quality
O_1	1	3	6	8
O_2	1	3	5	8
O_3	2	4	5	7
O_4	4	4	4	6
O_5	3	9	9	7
O_6	5	8	7	7



Skyline Context

Definition (Skyline Context)

- Let $\mathcal{O}$ be the space of all possible objects in space $\mathcal{D}$
- A *skyline context* is a triplet $(\mathcal{O}, 2^{\mathcal{D}}, R)$, where $R \subseteq \mathcal{O} \times 2^{\mathcal{D}}$ represents the skylines in subspaces

Skyline Context

Definition (Skyline Context)

- Let $\mathcal{O}$ be the space of all possible objects in space $\mathcal{D}$
- A *skyline context* is a triplet $(\mathcal{O}, 2^{\mathcal{D}}, R)$, where $R \subseteq \mathcal{O} \times 2^{\mathcal{D}}$ represents the skylines in subspaces

O	A	B	C	D	AB	AC	...	ABC	ABD	ACD	BCD	ABCD
$O_1^{=}$	X	X			X				X			
$O_1^{<>}$												
$O_2^{=}$	X	X			X				X			
$O_2^{<>}$						X		X		X	X	X
$O_3^{=}$												
$O_3^{<>}$									X	X		X
$O_4^{=}$												
$O_4^{<>}$			X	X		X		X	X	X	X	X

Closure Operators

Definition (Galois Connections)

A particular correspondence between the two partially ordered sets $2^{\mathcal{O}}$ and $2^{\mathcal{D}}$

- $\alpha(O) = \{\ \mathcal{U} \in 2^{\mathcal{D}} \mid SKY(\mathcal{U}) = O,\ \mathcal{U} \text{ is maximal }\}$
- $\beta(M) = \bigcap_{\mathcal{U} \in M} SKY(\mathcal{U})$

Definition (Skycube Closure)

$$h_* = \alpha \cdot \beta$$

The **closure** $h_*(M)$ of a set of subspaces M **returns the maximal subspaces** $\mathcal{U}$ having the **same skylines as** M

We have the partitioning of the skycube!

Closure Operators

Example

- $h_*(A) = \alpha \cdot \beta(A)$
- $\beta(A) = \{o_1^=, o_2^=\}$
- $\alpha(\{o_1^=, o_2^=\}) = \{AB\}$

O	**A**	**B**	**C**	**D**	**AB**	**AC**	**...**	**ABC**	**ABD**	**ACD**	**BCD**	**ABCD**
$O_1^{=}$	X	X			X				X			
$O_1^{<>}$												
$O_2^{=}$	X	X			X				X			
$O_2^{<>}$						X		X		X	X	X
$O_3^{=}$												
$O_3^{<>}$									X	X		X
$O_4^{=}$												
$O_4^{<>}$			X	X		X		X	X	X	X	X

Closure Operators

Example

- $h_*(A) = \alpha \cdot \beta(A)$
- $\beta(A) = \{o_1^=, o_2^=\}$
- $\alpha(\{o_1^=, o_2^=\}) = \{AB\}$

O	**A**	**B**	**C**	**D**	**AB**	**AC**	**...**	**ABC**	**ABD**	**ACD**	**BCD**	**ABCD**
$O_1^=$	X	X			X				X			
$O_1^{<>}$												
$O_2^=$	X	X			X				X			
$O_2^{<>}$						X		X		X	X	X
$O_3^=$												
$O_3^{<>}$									X	X		X
$O_4^=$												
$O_4^{<>}$			X	X		X		X	X	X	X	X

Closure Operators

Example

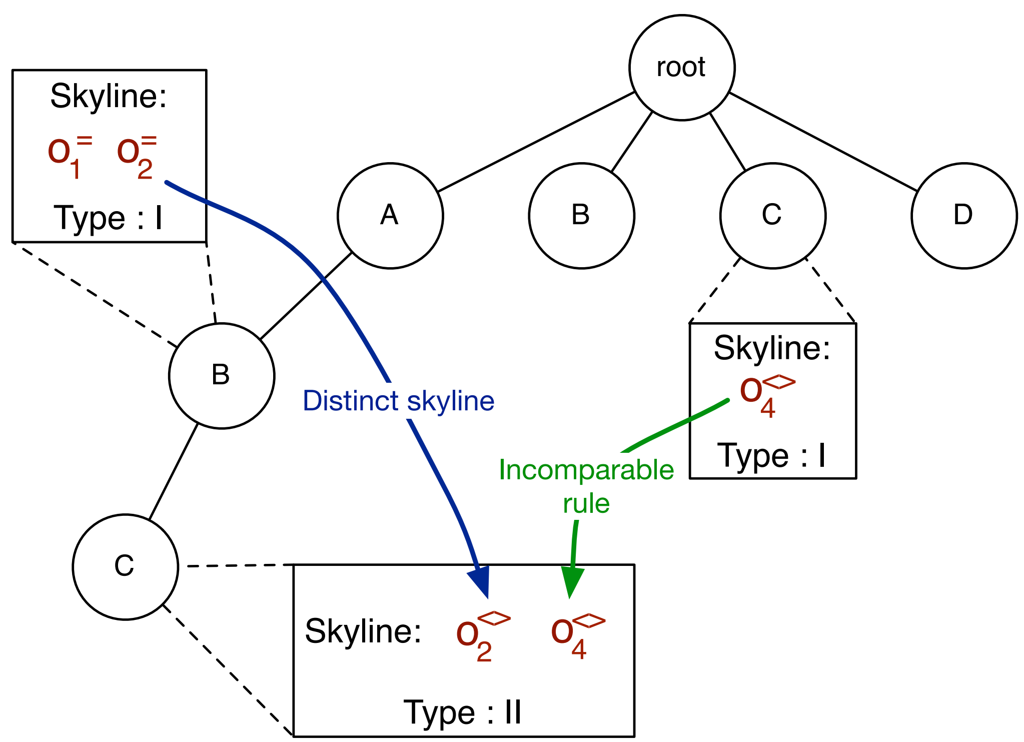
- $h_*(A) = \alpha \cdot \beta(A)$
- $\beta(A) = \{o_1^=, o_2^=\}$
- $\alpha(\{o_1^=, o_2^=\}) = \{AB\}$

O	A	B	C	D	AB	AC	...	ABC	ABD	ACD	BCD	ABCD
$O_1^=$	X	X			X				X			
$O_1^{<>}$												
$O_2^=$	X	X			X				X			
$O_2^{<>}$						X		X		X	X	X
$O_3^=$												
$O_3^{<>}$									X	X		X
$O_4^=$												
$O_4^{<>}$			X	X		X		X	X	X	X	X

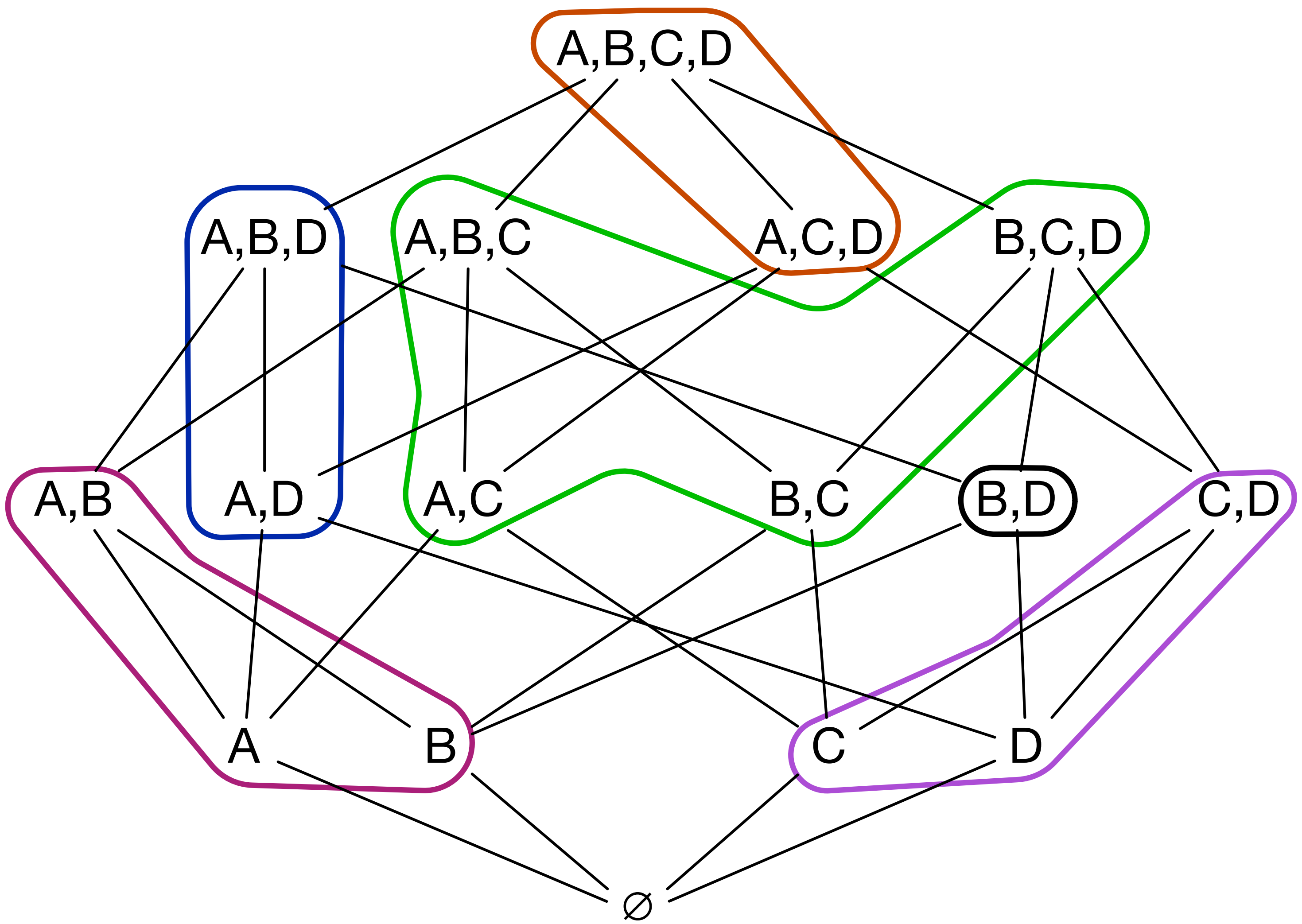
Orion-clos Algorithm

Basic Idea

- Use a prefix tree data structure
- Depth-first approach
- Use derivation rules
- Store the closed skycubes in a hash-list
- If the **processed subspace has the same skyline** than a previously computed closed skycube its branch may be **pruned**



Queries Over the Compressed Skycube



Closed	Sky	Generators
A,B,C,D	O2, O3, O4	A,C,D
A,B,D	O1, O2, O3, O4	A,D

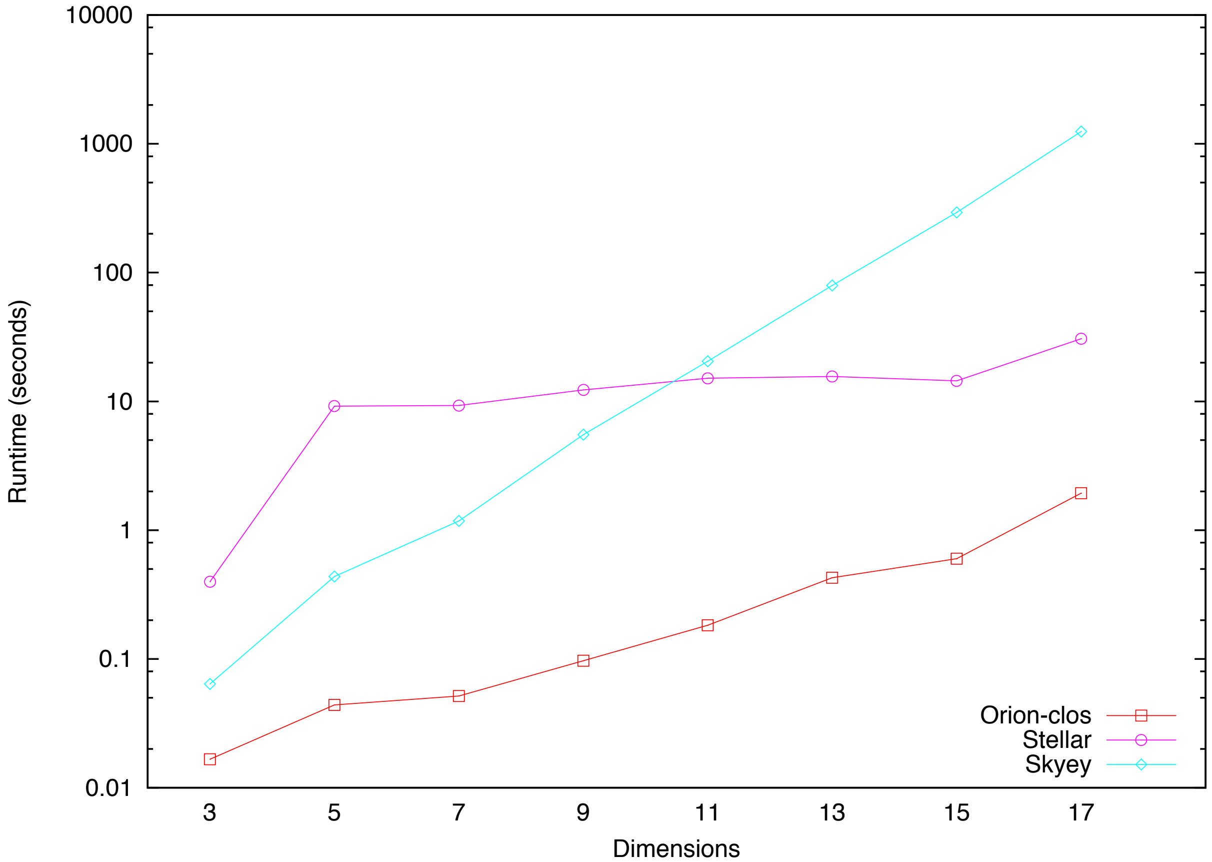
Linear time checks!

Experimental Setting

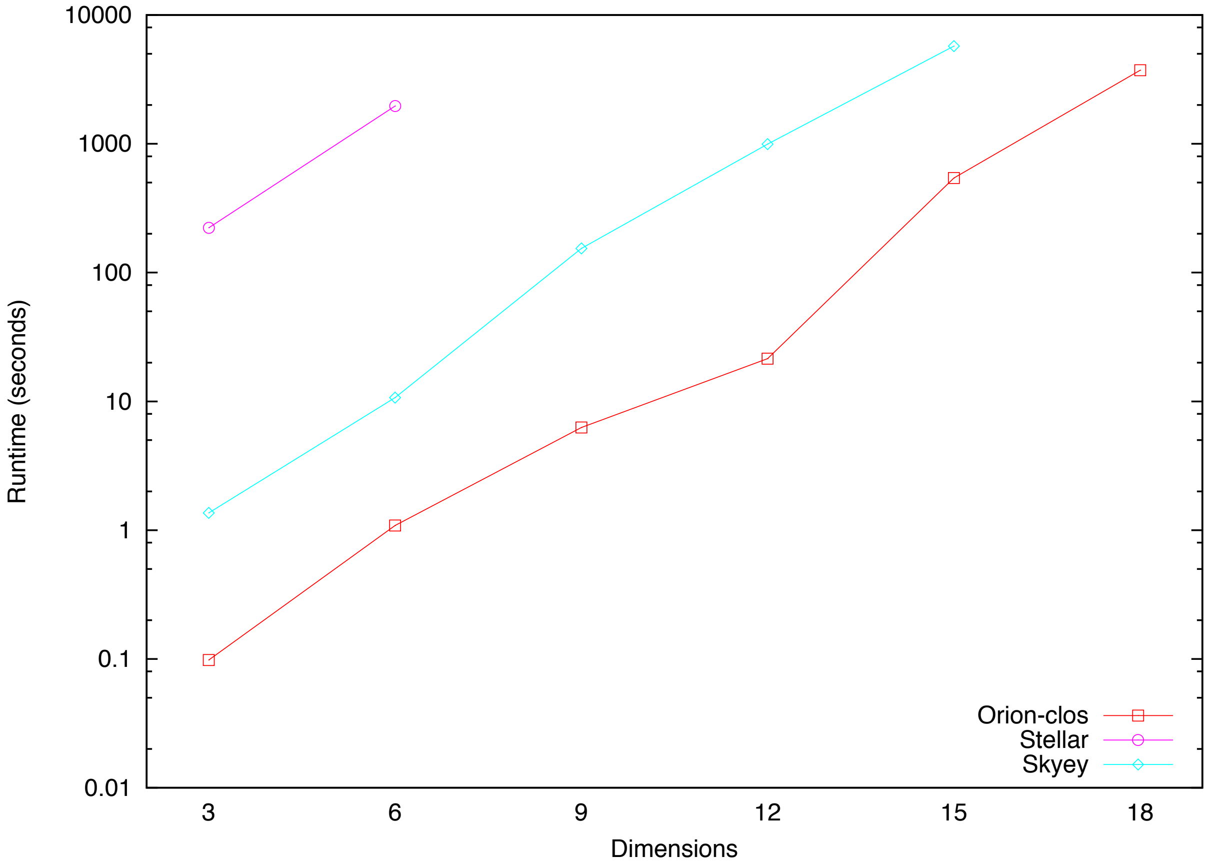
Algorithms
Orion
Orion-tail
Orion-clos
Stellar
Skyey

Data sets	Dimensionality	Objects
NBA	17	20493
MBL	18	92797
IPUMS	10	75836

Effect of Dimensionality

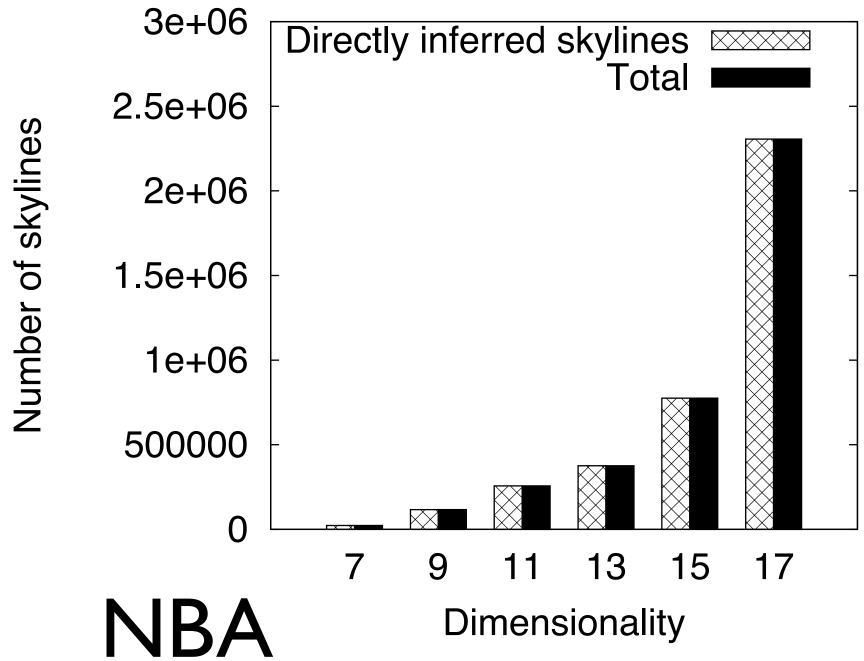


NBA

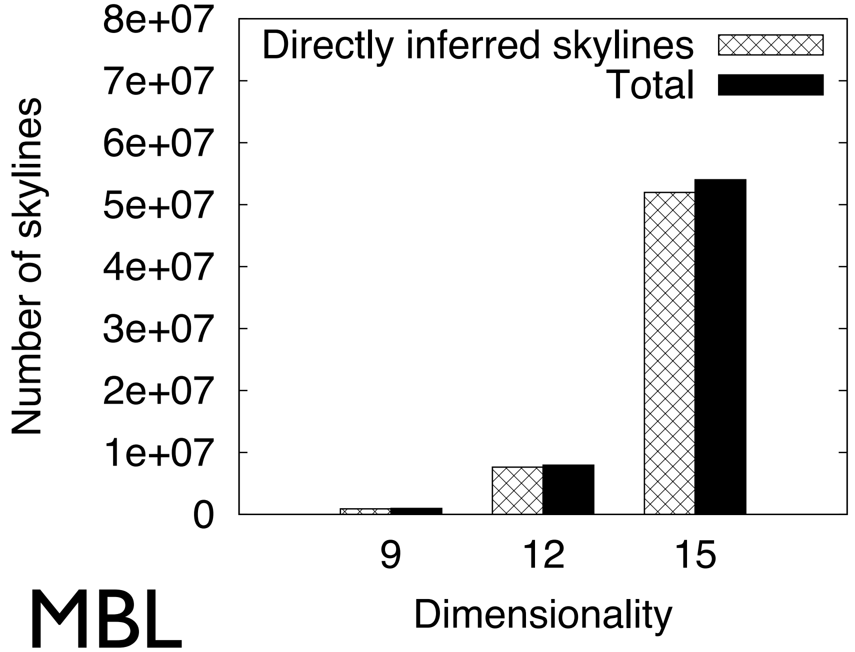


MBL

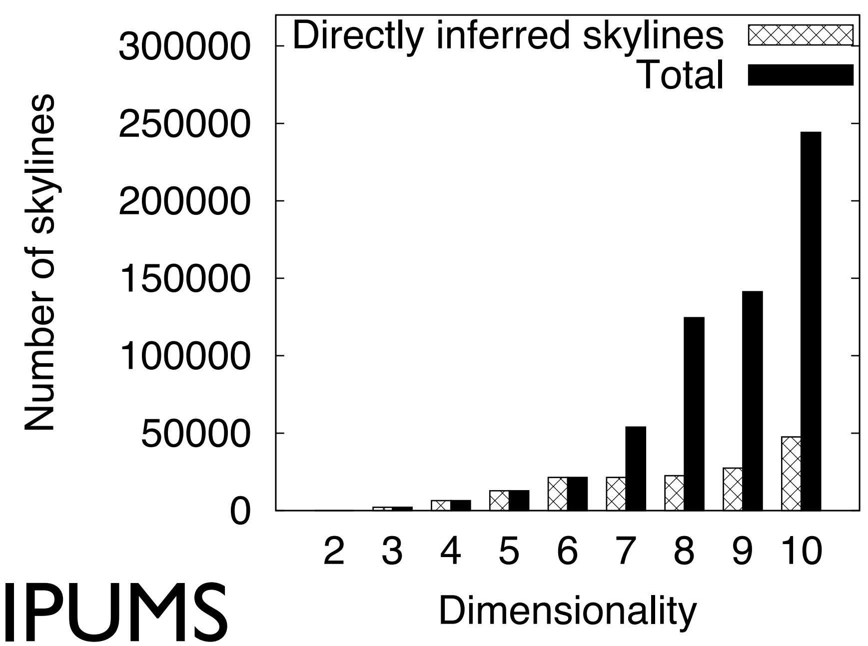
Skylines Inference



NBA

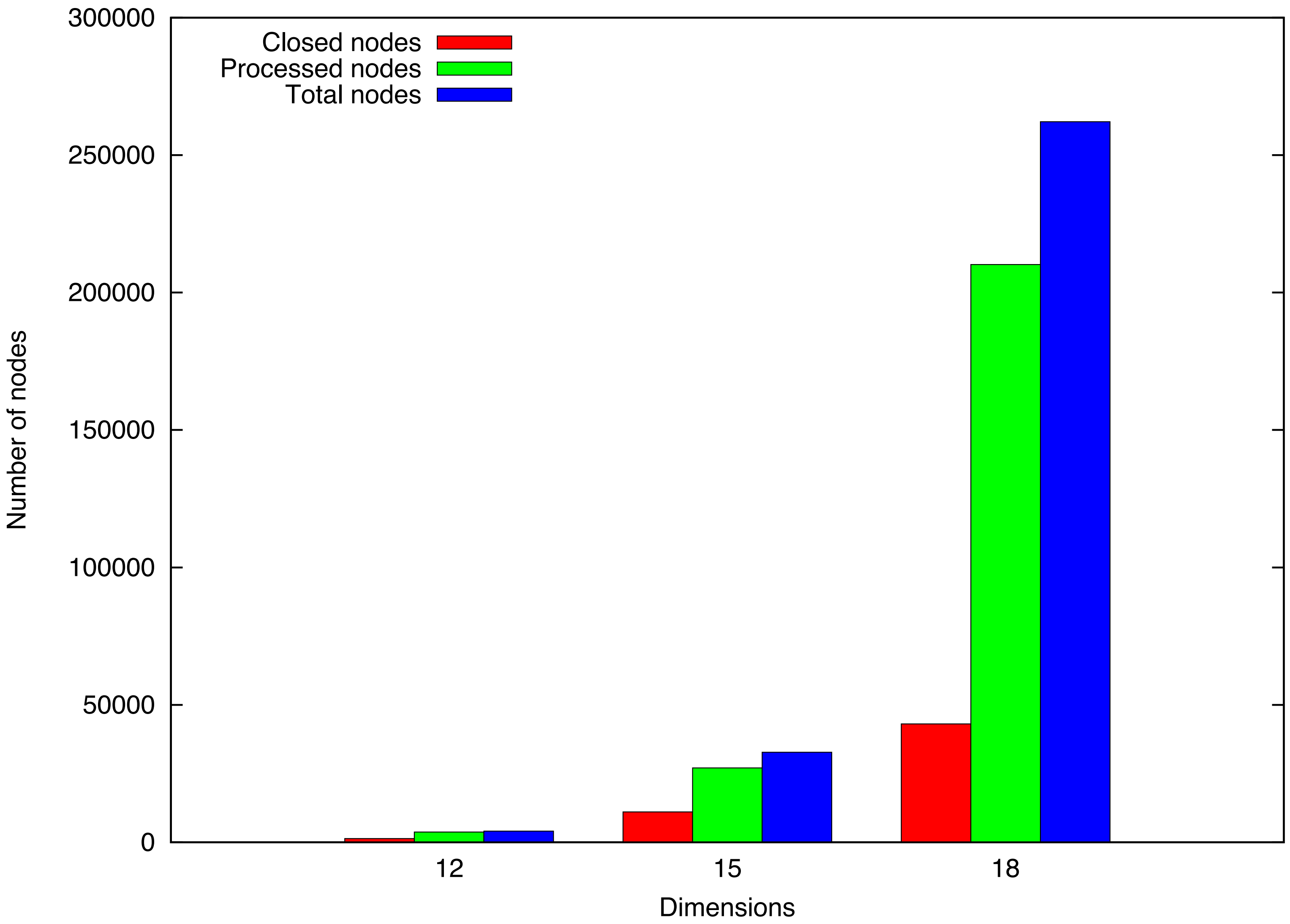


MBL



IPUMS

Number of Closed Skycubes



IPUMS data set

Summary

http://www.loria.fr/~raissi/
https://github.com/leander256/Orion

- A brief intro to the problem of Closed Skycubes Computation
- Different inference techniques (derivation rules) to quickly process skylines for different subspaces
- An efficient compression method is developed for skycubes
- Extensive performance evaluation show the superiority of Orion framework against related work

What is hot?

- Closed skycubes over data streams
- Querying the semantic web with preferences?

References

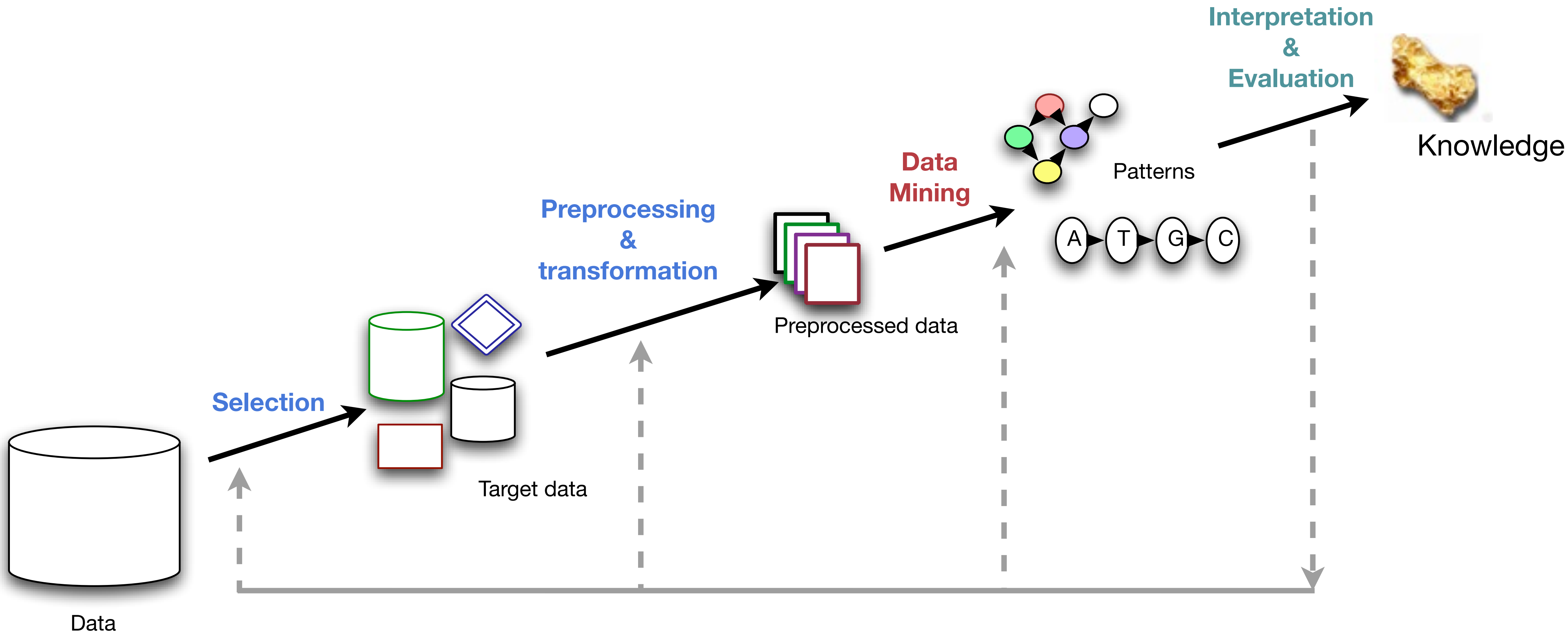
- *Efficient computation of the skyline cube*, Y. Yuan, X. Lin, Q. Liu, W. Wang, J. Xu Yu, and Q. zhang. [VLDB 2005]
- *Catching the best views of skyline: a semantic approach based on decisive subspaces*, J. Pei, W. Jin, M. Ester, and Y. Tao. [VLDB 2005]
- *Mining thick skylines over large databases*, W. Jin, J. Han, and M. Ester. [PKDD 2008]
- *Computing Closed Skycubes*, Chedy Raïssi, Jian Pei, and Thomas Kister. PVLDB 3(1): 838-847 (2010) [VLDB 2010]
- *Efficient parallel skyline processing using hyperplane projections*, H. Köhler, J. Yang, and X. Zhou [SIGMOD 2011]
- *Querying the semantic web with preferences*, W. Siberski, J. Z. Pan and U.Thaden [ISWC 2006]

4
SKYPATTERNS

Knowledge Discovery in Databases

Knowledge Discovery in Databases (KDD) revolves around the investigation and creation of knowledge, processes, algorithms, and the mechanisms for **retrieving potential knowledge** from data collections

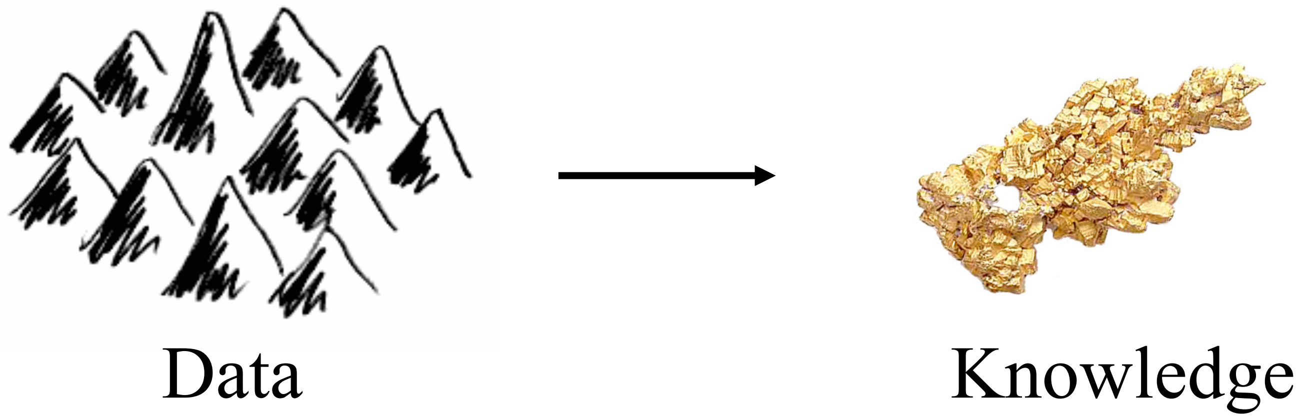
Knowledge Discovery in Databases



Knowledge Discovery in Databases (KDD) revolves around the investigation and creation of knowledge, processes, algorithms, and the mechanisms for **retrieving potential knowledge** from data collections

Data Mining

... is "the use of sophisticated data analysis tools to **discover** previously unknown, **valid patterns and relationships** in large data sets"



Useful for...

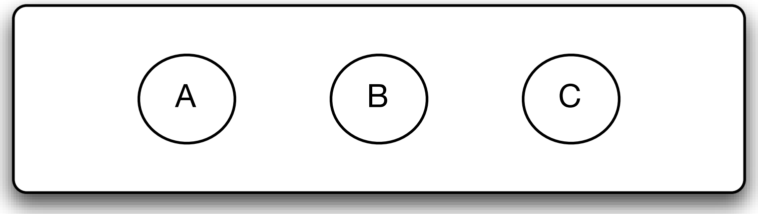
- Classification
- Clustering
- Prediction
- Correlation analysis

Pattern Mining

- Patterns are subclasses of directed graphs
- Patterns **depend** on the **data type** and the **applications needs**

Pattern Mining

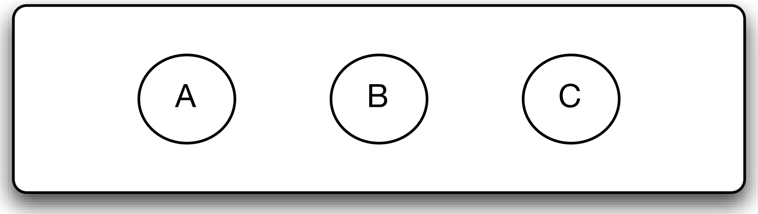
- Patterns are subclasses of directed graphs
- Patterns **depend** on the **data type** and the **applications needs**



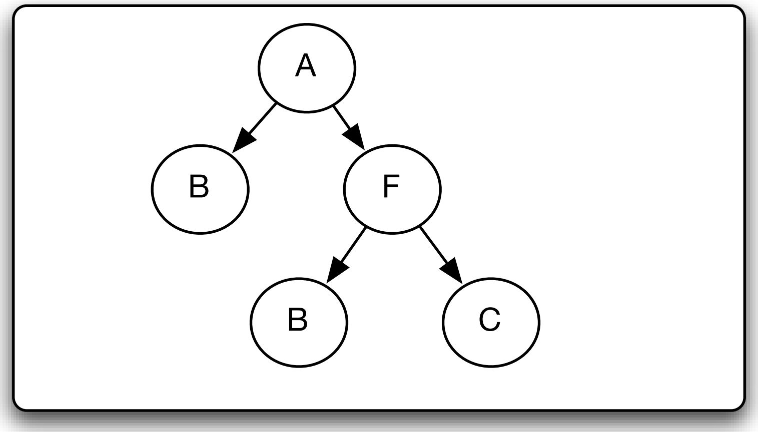
Itemset

Pattern Mining

- Patterns are subclasses of directed graphs
- Patterns **depend** on the **data type** and the **applications needs**



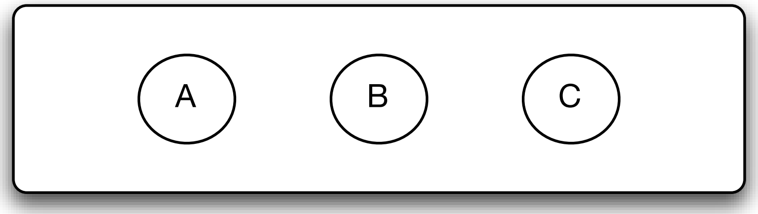
Itemset



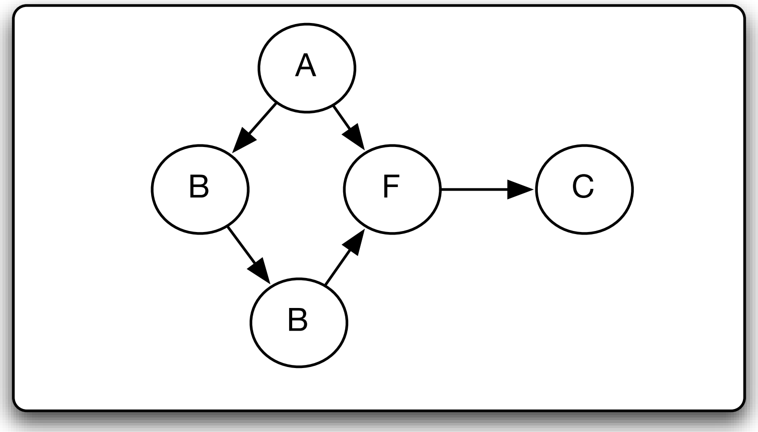
Tree

Pattern Mining

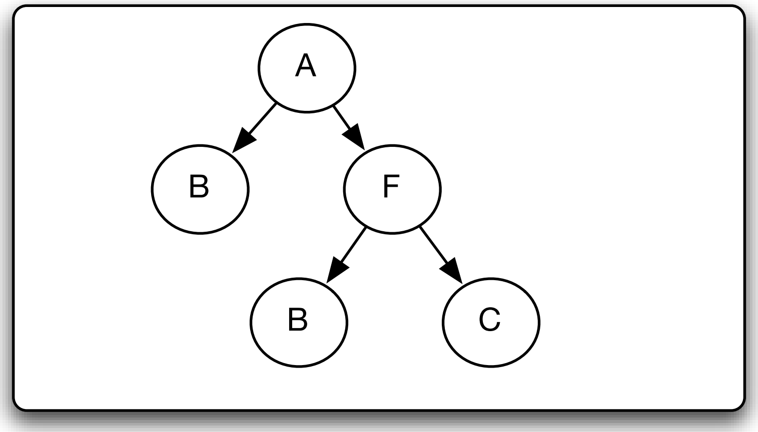
- Patterns are subclasses of directed graphs
- Patterns **depend** on the **data type** and the **applications needs**



Itemset



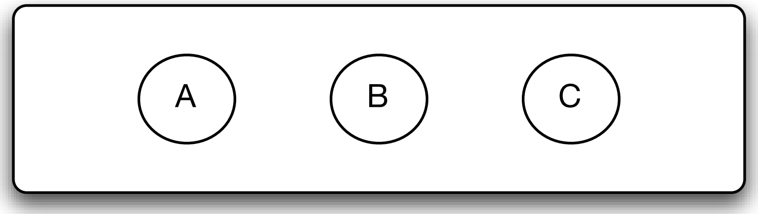
Graph



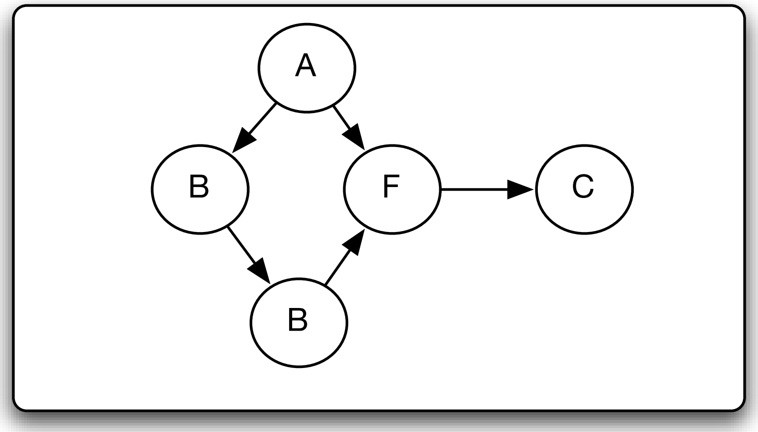
Tree

Pattern Mining

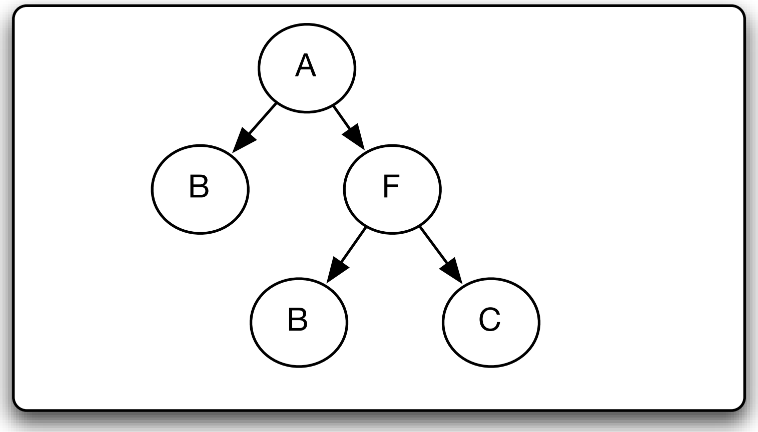
- Patterns are subclasses of directed graphs
- Patterns **depend** on the **data type** and the **applications needs**



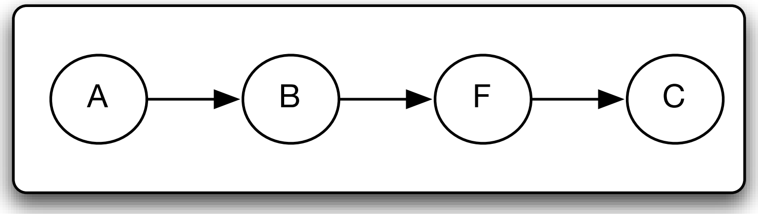
Itemset



Graph



Tree



Sequence

Transaction Data Analysis

- Introduced in [Agrawal, Imielinski and Swami, SIGMOD 1993]
- Enables the discovery of **correlations** between **items**

- Transactions: customers' purchases of commodities
  - $\{bread, wine, cheese\}$ bought together
- Frequent patterns: product combinations that are frequently purchased together
- Frequent patterns: patterns (set of items, sequences, graphs) that occur frequently in a database

Frequent Itemset Mining

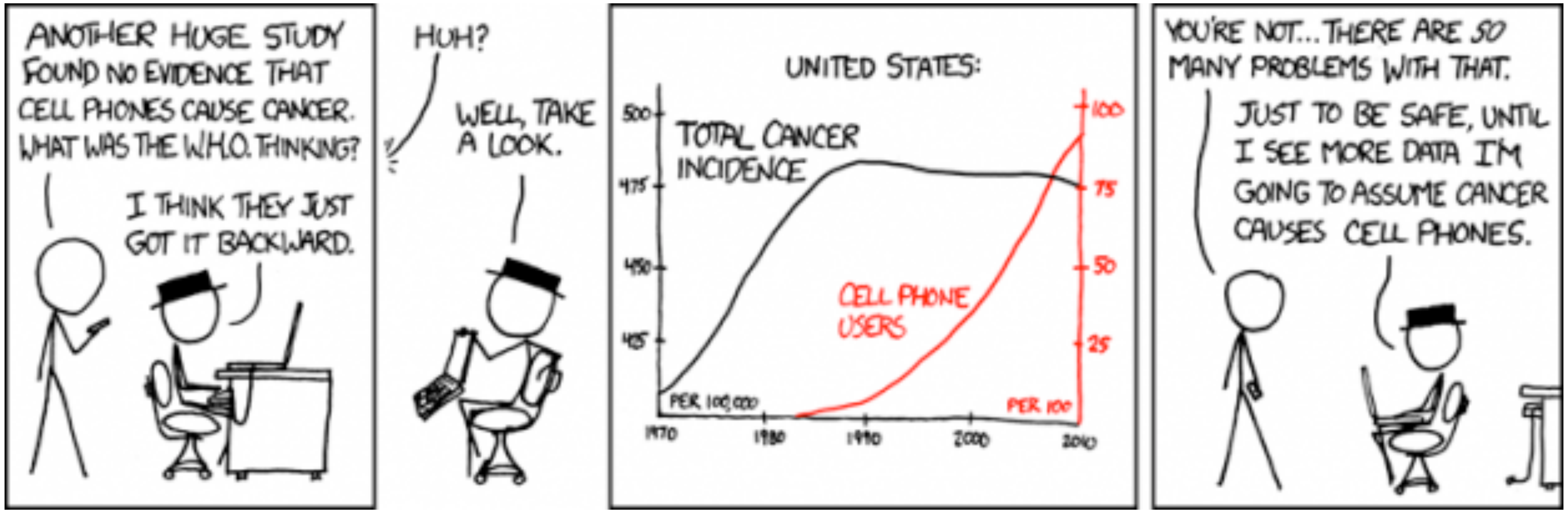
- Itemset: a set of items, e.g $\{a, c, m\}$
- Support of itemsets $Sup(acm) = 3$
- Frequent pattern mining is an enumeration problem given a minimal support constraint ϵ

Transaction database TDB

TID	Items bought
100	f, a, c, d, g, I, m, p
200	a, b, c, f, l, m, o
300	b, f, h, j, o
400	b, c, k, s, p
500	a, f, c, e, l, p, m, n

Frequent Itemset Mining

- Itemset: a set of items, e.g $\{a, c, m\}$
- Support of itemsets $Sup(acm) = 3$
- Frequent pattern mining is an enumeration problem given a minimal support constraint ϵ



Motivations

- Discovering patterns satisfying a global property
  Dominance relation
- Give the end-user a new and easy way to express his preferences
  In a multidimensional space: each dimension is a measure
- Avoid the threshold issue
  What is the *"best"* value of my minimal frequency ?
  What k in top-k?
  Combining several measures ?

What if it also gives a way to discover less (and promising) patterns ?

Notion of skyline patterns

The basic idea: if a pattern is dominated by another according to all measures in a set M then it is discarded in the output.
($X \succ_M Y$: X dominates Y)

Let P be a pattern set. A **skypattern** of P with respect to M is a pattern not dominated in P with respect to M.

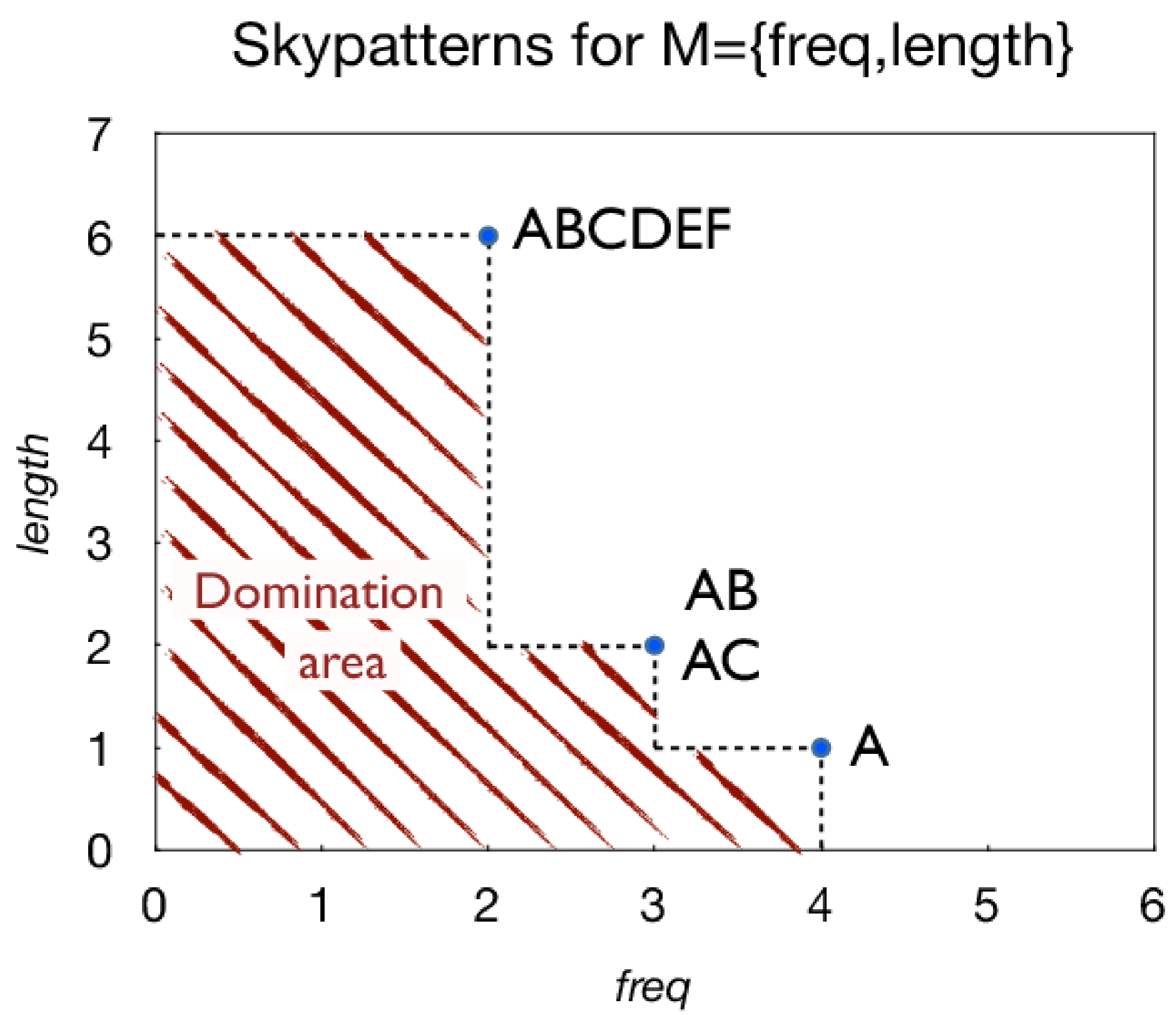
The **skypattern operator** $\mathcal{S}ky(P, M)$: returns all the skypatterns of P with respect to M:

$$\mathcal{S}ky(P, M) = \{X \in P \mid \nexists Y \in P : Y \succ_M X\}$$

Example

Tid	Items
t_1	A B C D E F
t_2	A B C D E F
t_3	A B
t_4	D
t_5	A C
t_6	E

Patterns	**freq**	**length**
$ABCDEF$	2	6
AB	3	2
AC	3	2
A	4	1



$$Sky(\mathcal{L}, \{freq, length\}) = \{ABCDEF, AB, AC, A\}$$

Algorithmic Issues

A naive enumeration of $\mathcal{L}$ (the whole set of possible patterns) and then a comparison between the patterns is not possible.

Key idea: Take benefit from the pattern condensed representation according to the condensable measures of M.

Skylineability

1. **Skylineability**: Look for a smaller set of measures M' from M with interesting properties on the skypatterns.

   M is M'-**skylineable** with respect to $\subset$ (resp. $\supset$) iff for any patterns $X =_{M'} Y$ such that $X \subset Y$ (resp. $X \supset Y$), one has $X \succeq_M Y$.

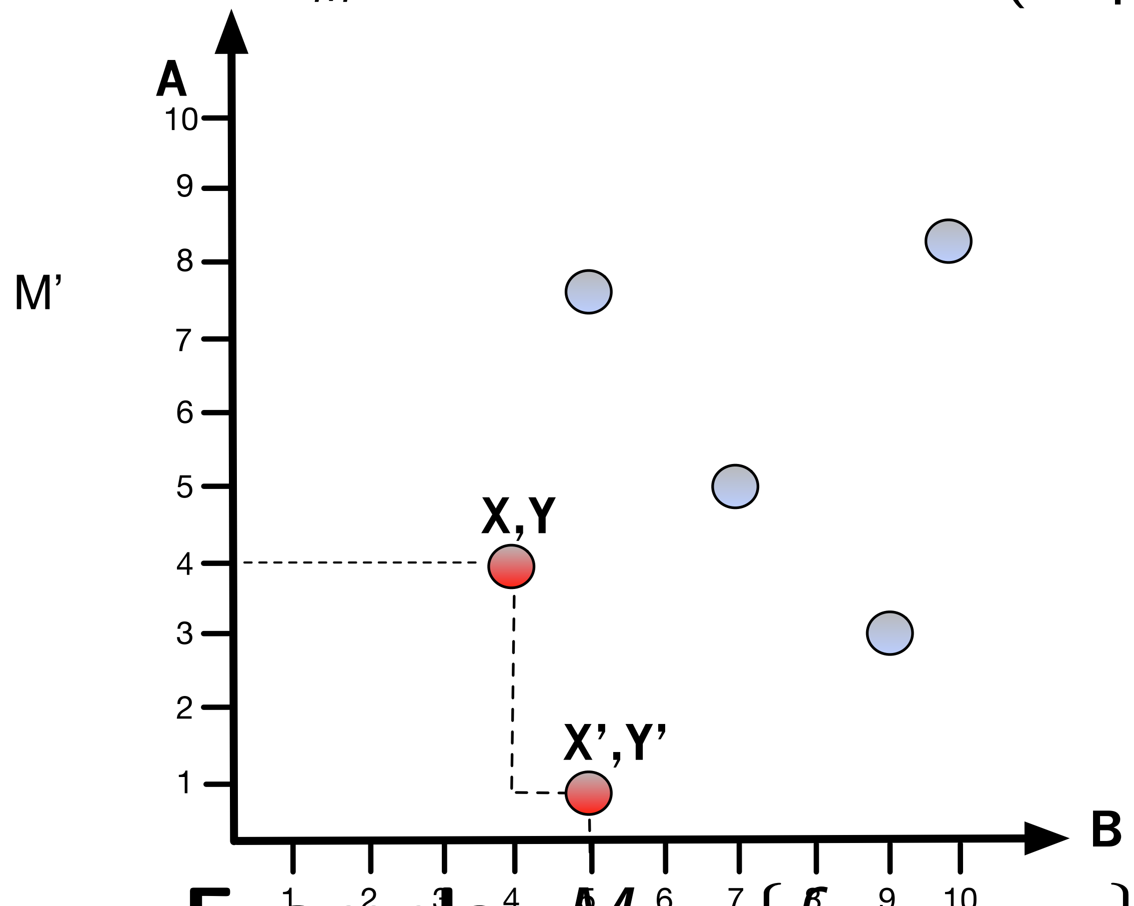
Example: $M = \{freq, area\}$ is strictly $\{freq\}$-skylineable with respect to $\supset$.

$B =_{freq} AB$: we can directly deduce that $AB \succ_M B$.

Skylineability

1. **Skylineability**: Look for a smaller set of measures M' from M with interesting properties on the skypatterns.

   M is M'-**skylineable** with respect to $\subset$ (resp. $\supset$) iff for any patterns $X =_{M'} Y$ such that $X \subset Y$ (resp. $X \supset Y$), one has $X \succeq_M Y$.



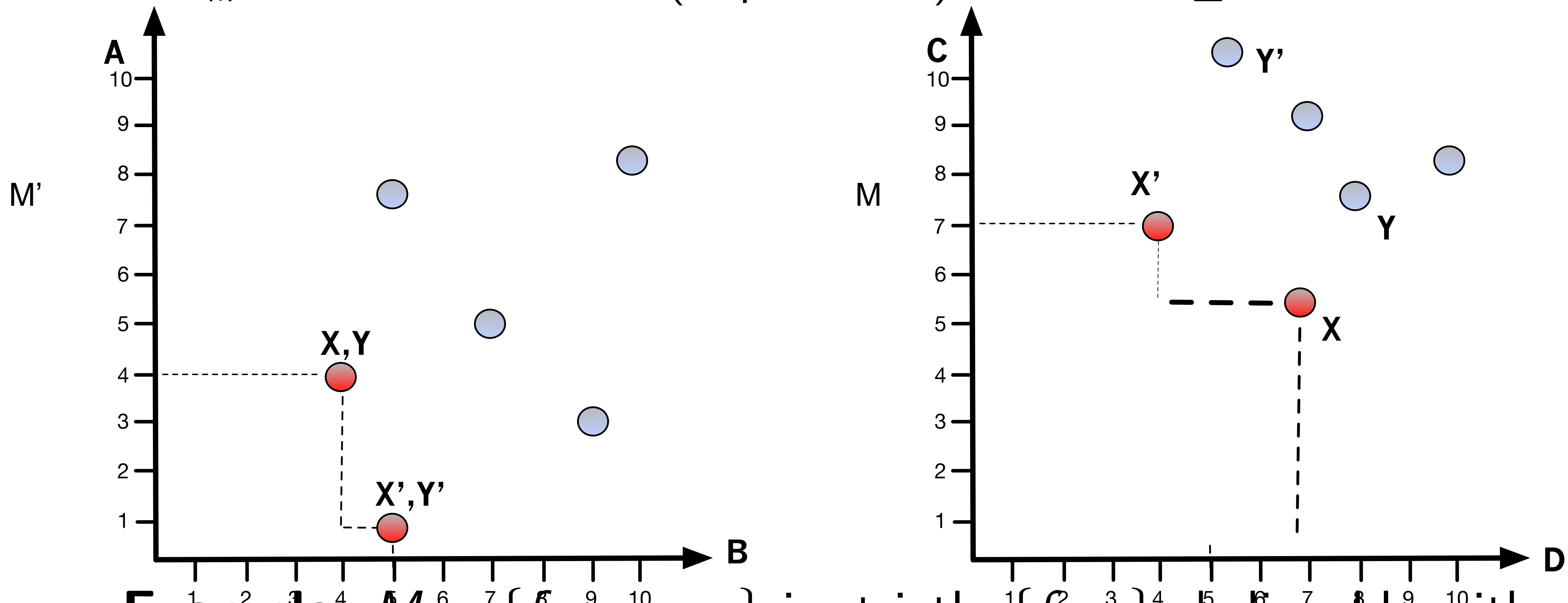
Example: $M = \{freq, area\}$ is strictly $\{freq\}$-skylineable with respect to $\supset$.

$B =_{freq} AB$: we can directly deduce that $AB \succ_M B$.

Skylineability

1. **Skylineability**: Look for a smaller set of measures M' from M with interesting properties on the skypatterns.

   M is M'-**skylineable** with respect to $\subset$ (resp. $\supset$) iff for any patterns $X =_{M'} Y$ such that $X \subset Y$ (resp. $X \supset Y$), one has $X \succeq_M Y$.



Example: $M = \{freq, area\}$ is strictly $\{freq\}$-skylineable with respect to $\supset$.

$B =_{freq} AB$: we can directly deduce that $AB \succ_M B$.

Computing automatically M'

1. Minimizer and maximizer operators to compute M'.
2. Done using a syntax tree.

Computing automatically M'

1. Minimizer and maximizer operators to compute M'.
2. Done using a syntax tree.

Expr. e	Primitive(s)	$\underline{c}(e)$	$\overline{c}(e)$
$e_1 \theta e_2$	$\theta \in \{+, \times, \cup\}$	$\underline{c}(e_1) \cup \underline{c}(e_2)$	$\overline{c}(e_1) \cup \overline{c}(e_2)$
$e_1 \theta e_2$	$\theta \in \{-, /, \cap\}$	$\underline{c}(e_1) \cup \overline{c}(e_2)$	$\overline{c}(e_1) \cup \underline{c}(e_2)$
constant	-	$\emptyset$	$\emptyset$
$d(X)$	$d \in \{freq, min, g\}$	$\emptyset$	$\{d(X)\}$
$i(X)$	$i \in \{length, max, sum, freq_\vee, f\}$	$\{i(X)\}$	$\emptyset$
$d(e_1)$	$d \in \{freq, min, g\}$	$\overline{c}(e_1)$	$\underline{c}(e_1)$
$i(e_1)$	$i \in \{length, max, sum, freq_\vee, f\}$	$\underline{c}(e_1)$	$\overline{c}(e_1)$

Computing automatically M'

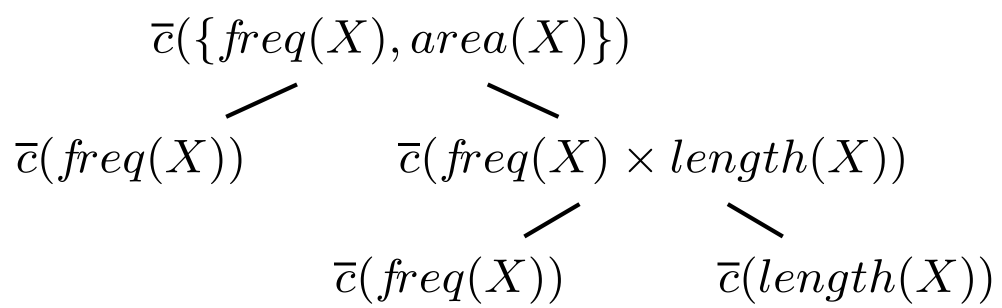
1. Minimizer and maximizer operators to compute M'.
2. Done using a syntax tree.

Expr. e	Primitive(s)	$\underline{c}(e)$	$\overline{c}(e)$
$e_1 \theta e_2$	$\theta \in \{+, \times, \cup\}$	$\underline{c}(e_1) \cup \underline{c}(e_2)$	$\overline{c}(e_1) \cup \overline{c}(e_2)$
$e_1 \theta e_2$	$\theta \in \{-, /, \cap\}$	$\underline{c}(e_1) \cup \overline{c}(e_2)$	$\overline{c}(e_1) \cup \underline{c}(e_2)$
constant	-	$\emptyset$	$\emptyset$
$d(X)$	$d \in \{freq, min, g\}$	$\emptyset$	$\{d(X)\}$
$i(X)$	$i \in \{length, max, sum, freq_\vee, f\}$	$\{i(X)\}$	$\emptyset$
$d(e_1)$	$d \in \{freq, min, g\}$	$\overline{c}(e_1)$	$\underline{c}(e_1)$
$i(e_1)$	$i \in \{length, max, sum, freq_\vee, f\}$	$\underline{c}(e_1)$	$\overline{c}(e_1)$

(a) Individual measures

Meas. m	$\overline{c}(m)$	$\underline{c}(m)$
$area$	$\{freq(X)\}$	$\{length(X)\}$
$mean$	$\{min(X.val)\}$	$\{max(X.val)\}$
$bond$	$\{freq(X), freq_\vee(X)\}$	$\emptyset$
$aconf$	$\{freq(X), max(X.val)\}$	$\emptyset$
gr_1	$\{freq(X, \mathcal{D}_1)\}$	$\{freq(X, \mathcal{D}_2)\}$

(b) A set of measures $M = \{freq(X), area(X)\}$



Computing Concise representations according to M'

$$\mathcal{D}is_{\theta}(P, M') = \{X \in P | \forall Y \theta X : X \neq_{M'} Y\} \text{ where } \theta \in \{\subset, \supset\}$$

The distinct operation for $P \subseteq \mathcal{L}$ with respect to M' and $\theta \in \{\subset, \supset\}$ returns all the patterns X of P such that their generalizations (or specializations) are distinct from X with respect to M'

(a) A toy data set $\mathcal{D}$

Tid	Items					
t_1	A	B	C	D	E	F
t_2	A	B	C	D	E	F
t_3	A	B				
t_4				D		
t_5	A		C			
t_6					E	

Example:
$\mathcal{D}is_{\subset}(\mathcal{L}, \{freq\}) = \{A, B, C, D, E, F, AD, AE, BC, BD, BE, CD, CE, DE\}$
and $\mathcal{D}is_{\supset}(\mathcal{L}, \{freq\}) = \{A, D, E, AB, AC, ABCDEF\}$.

Aetheris Approach

1. Compute the best M'
2. Process distinct patterns given M'
3. Compute the skyline patterns from the condensed representation

4. Finalize by generating all the skypatterns: retrieving of all the indistinct patterns from their representatives

$$\mathcal{I}nd(\mathcal{L}, M', P) = \{X \in \mathcal{L} | \exists Y \in P : X =_{M'} Y\}$$

Example: $\mathcal{I}nd(\mathcal{L}, \{freq\}, \{AB, AC\}) = \{B, C, AB, AC\}$

Finally:

$$\mathcal{S}ky(\mathcal{L}, M) = \mathcal{I}nd(\mathcal{L}, M, \mathcal{S}ky(\mathcal{D}is_\theta(\mathcal{L}, M'), M))$$

Aetheris Approach: Example

(a) A toy data set $\mathcal{D}$

Tid	Items					
t_1	A	B	C	D	E	F
t_2	A	B	C	D	E	F
t_3	A	B				
t_4				D		
t_5	A		C			
t_6					E	

4. $\mathcal{I}nd$ $= \{A, AB, AC, ABCDEF\}$
3. $\mathcal{L}$ $\quad \{freq; area\}$ $\quad \mathcal{S}ky$ $= \{A, AB, AC, ABCDEF\}$
2. $\{freq; area\}$ $\quad \mathcal{D}is_{\supset}$ $= \{A, D, E, AB, AC, ABCDEF\}$
1. $\mathcal{L}$ $\quad \overline{c}(\{freq; area\})$ $= \{freq\}$

Figure 3: Computing the skypatterns with respect to $\{freq; area\}$ from running example

Experiments on Itemset Data ($\mathcal{L} = 2^{\mathcal{I}}$)

Experiments on UCI data

- 16 benchmarks.
- Synthesis of 128 experiments.
- Runtimes only consider the application of skyline operator.

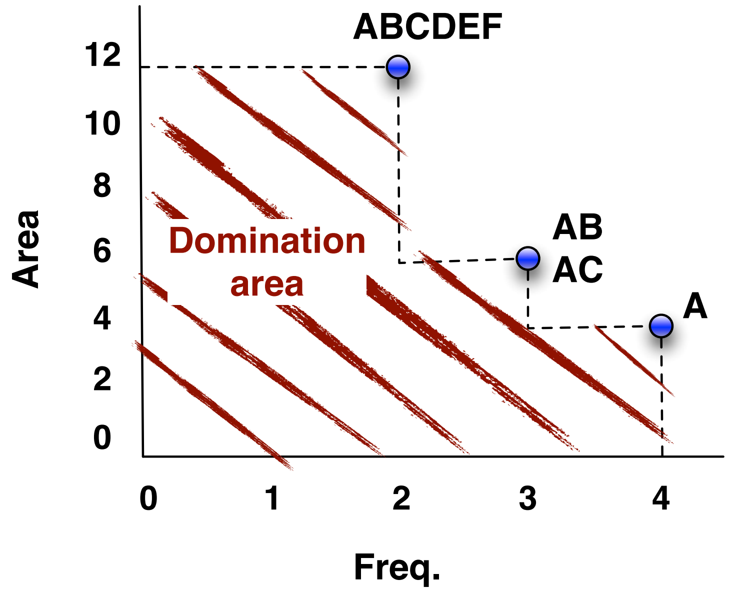
Comparisons of 3 approaches

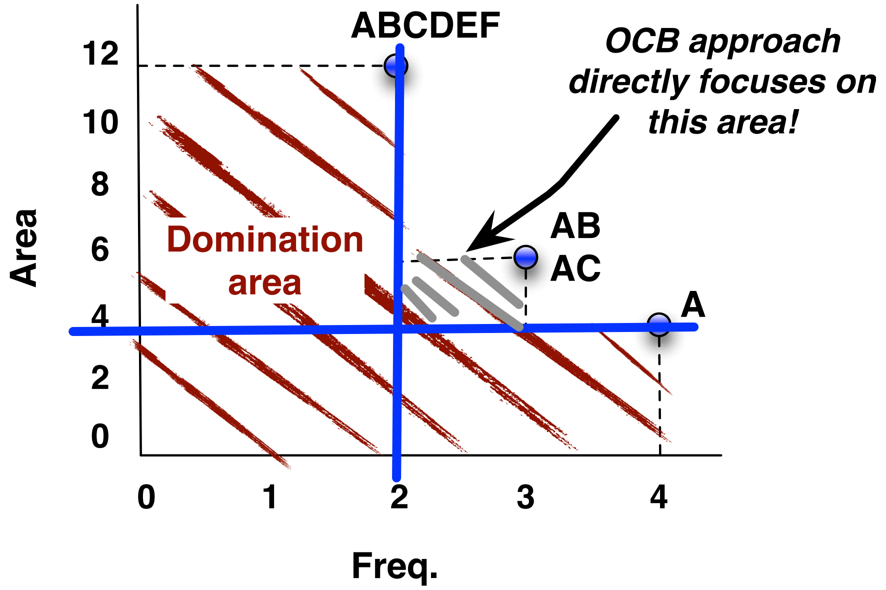
1. **Baseline approach:** $\mathcal{S}ky\ (\{X \subseteq \mathcal{I} \mid freq(X, \mathcal{D}) \geq 1\}, M)$.
2. **Optimal Constraint-Based approach:** Assume that user set the *optimal* thresholds
3. **Aetheris approach:**
   $\mathcal{S}ky(\mathcal{L}, M) = \mathcal{I}nd(\mathcal{L}, M, \mathcal{S}ky(\mathcal{D}is_\theta(\mathcal{L}, M'), M))$

Optimal Constraint-Based Approach Settings in a Nutshell

Tid	Items
t_1	A B C D E F
t_2	A B C D E F
t_3	A B
t_4	D
t_5	A C
t_6	E

Patterns	**freq**	**length**	**area**
$ABCDEF$	2	6	12
AB	3	2	6
AC	3	2	6
A	4	1	4



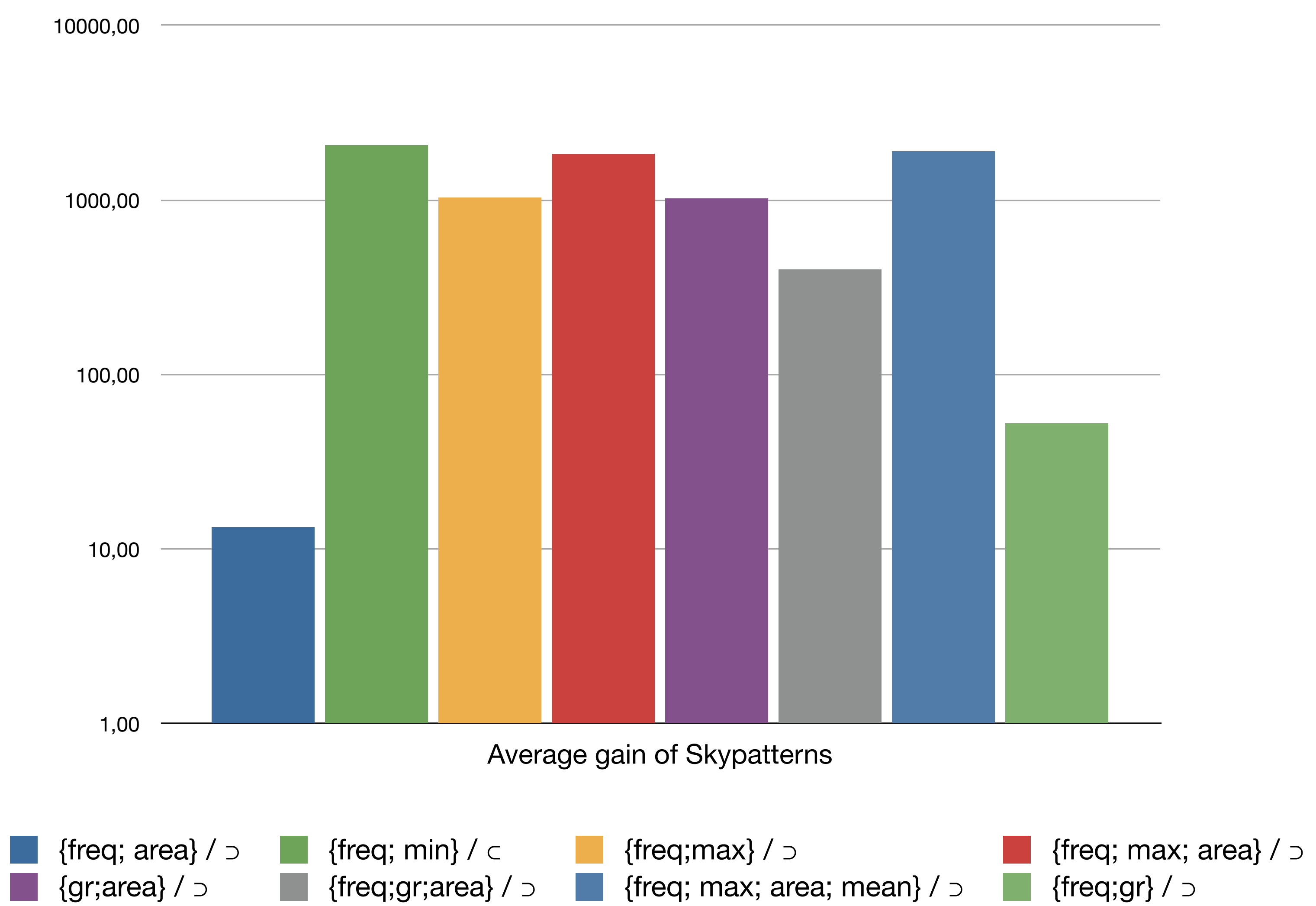


$$Sky(\mathcal{L}, \{freq, area\}) = \{ABCDEF, AB, AC, A\}$$

$$\sigma_{sup} = 2 \text{ and } \sigma_{area} = 4$$

Results: Conciseness Gain

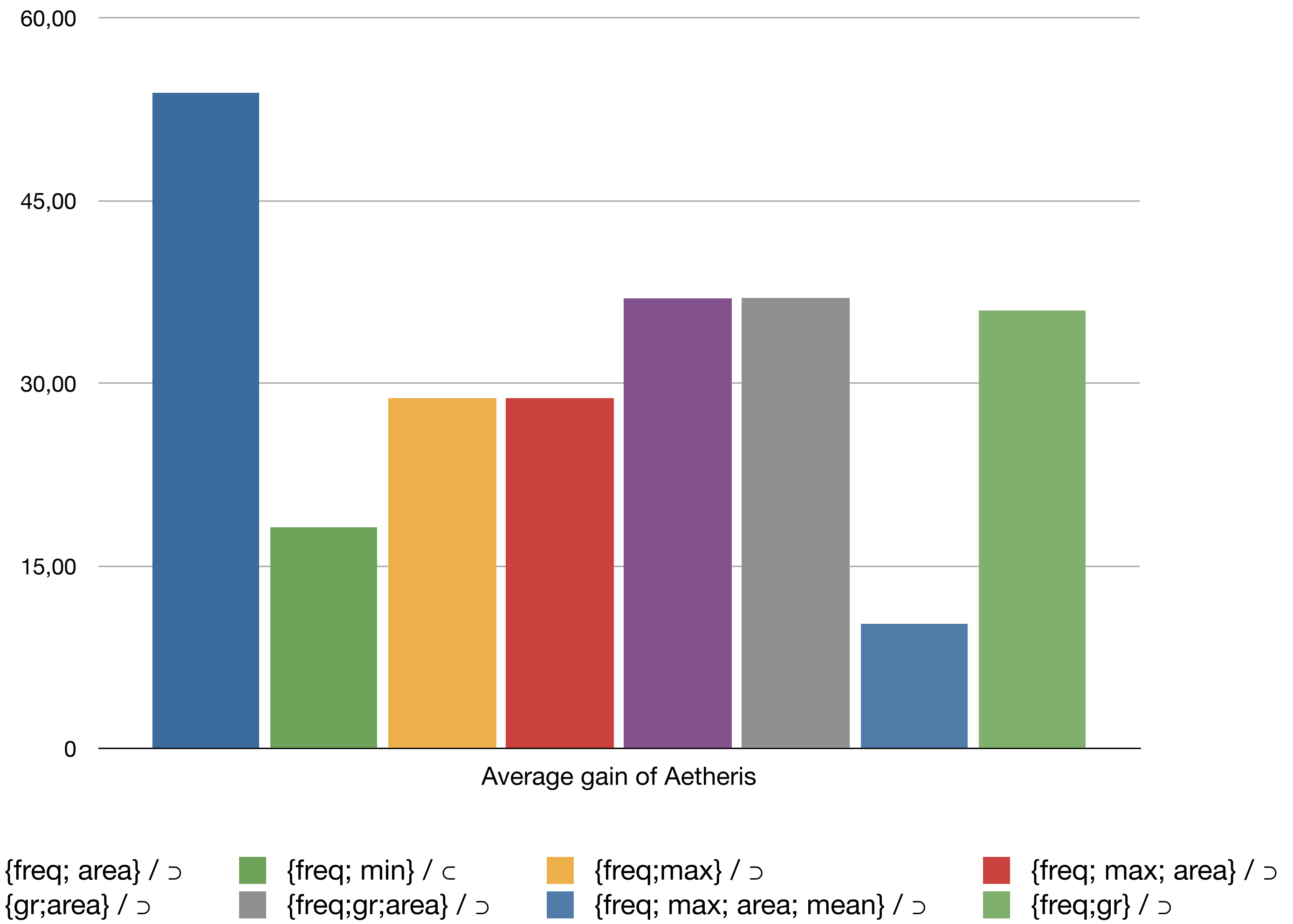
Average gain of skypatterns according to OCB patterns



The gain of a skyline approach is always important (greater than 10 and much greater in almost all the cases).

Results: Performance Gain

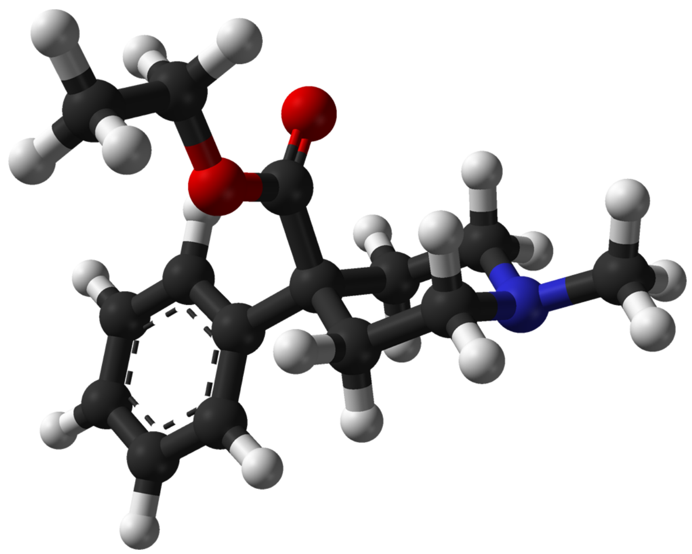
Runtime gain of Aetheris according to Baseline



Aetheris always outperforms the baseline approach with at least a factor of 10.

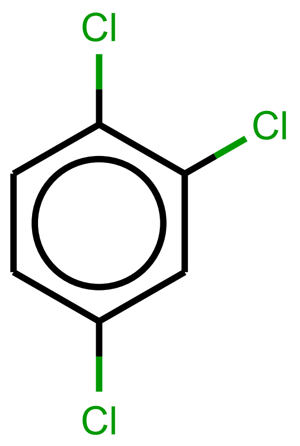
Case Study: Discovering Toxicophores

- Collaboration with the CERM Laboratory.
- Establishing relationships between chemicals and (eco)toxicity



Our aim: Investigate the use of skypatterns to discover toxicophores

ECB[a] Dataset: 567 chemicals (372 very toxic/195 harmful)



trichlorobenzene

---

[a]European Chemicals Bureau
`http://echa.europa.eu/`

Case study: Results

Experiment 1: contrast measures (e.g., growth rate) are useful to discover toxicophores

- only 8 skypatterns!
- the method is able to automatically discover already known environmental toxicophores:
  ➥ it suggests good insights for the others

Experiment 2: background knowledge can easily be integrating adding aromaticity and density measures

- the whole set of skypatterns remains small (38 skypatterns)
- discovering of skypatterns including an amine function not detected in Experiment 1

Summary

- A novel pattern mining problem.
- Useful results from a *user-preference* point of view.
- No thresholds $\longrightarrow$ Threshold-free constraint based pattern mining is possible!
- Use case: Discovering toxicophores

Future Work

- Devise new pruning strategy like *"approximate and push"*.
- Apply it on more complex patterns (sequence, graph, sequence of graphs).
- Simultaneously on several languages (itemset, sequence, etc.).
- Enabling full interactivity:
  - Deleting some skypatterns.
  - Adding/deleting some preferences $\rightarrow$ skypattern cube computation.
- Skypatterns and background knowledge: see Szimon's or M. Van Leuwen's papers.
- Skypatterns-based classification.
- Skypattern and their covering.

References

- *SkyGraph: An Algorithm for Important Subgraph Discovery in Relational Graphs*, A. N. Papadopoulos, A. Lyritsis, and Y. Manolopoulos. [PKDD2008]
- *Mining Dominant Patterns in the Sky*, A. Soulet, C. Raïssi, M. Plantevit, B. Crémilleux. [ICDM2011]

THANK YOU

QUESTIONS?