Pattern Sets

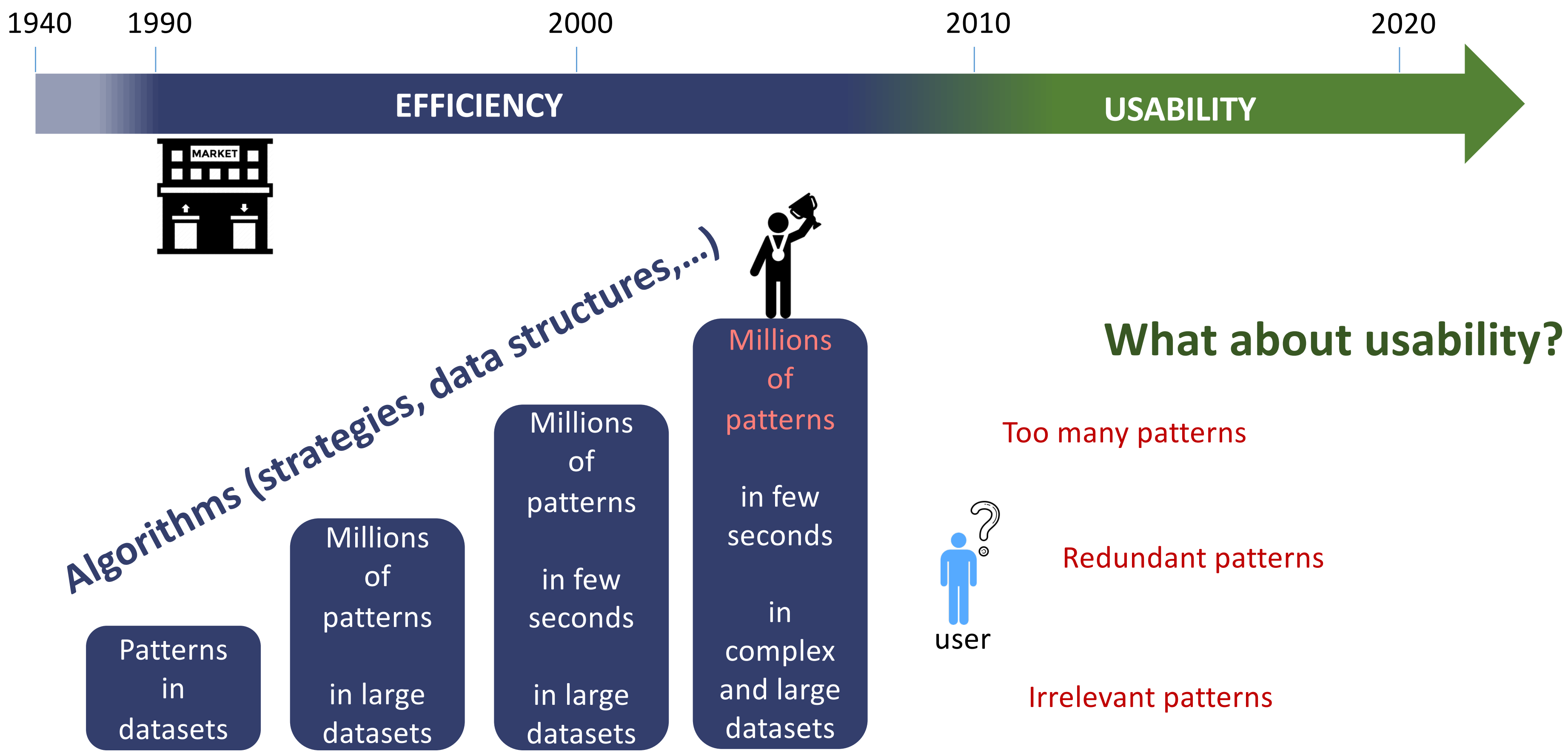
(DMV Lecture, M2 SIF)

Peggy Cellier

peggy.cellier@irisa.fr

Last revision: October 2023

Pattern Mining: Efficiency, Efficiency, Efficiency



How to manage the amount of Patterns?

Most of
measure the interest of
each pattern individually

measure the interest of
the set of patterns

Condensed representations

Examples
- Closed patterns
  [Pasquier+'99]
- Maximal patterns

Constraints

Examples
- Length of the patterns
- Regular expressions
  [Garofalakis+'99]

Measures of interest

Examples [Geng+'06]
- Support
- Confidence
- Growth rate

Pattern sets

Examples
- Tiles [Geerts+'04]
- **Compression-based**
  [Vreeken+'11]
- Swap randomization
  [Lijffijt+'12]

References and course material

- [1] Krimp: mining itemsets that compress, Jilles Vreeken, Matthijs van Leeuwen, Arno Siebes, DMKD 2011
- [2] Slim: Directly Mining Descriptive Patterns, Koen Smets and Jilles Vreeken, SDM 2012
- [3] The Long and the Short of It: Summarising Event Sequences with Serial Episodes, Nikolaj Tatti and Jilles Vreeken, KDD 2012
- [4] The Minimum Description Length Principle, Peter Grünwald, MIT Press, 2007

- Some (lot of) slides come from the talk of Jilles Vreeken
  - https://people.mmci.uni-saarland.de/~jilles/
- Slides from the HDR of Peggy Cellier

Outline

1. **Introduction**
2. **Compression based approaches**
    1. Motivations
    2. Information theory and the MDL principle
        - Information theory
        - The MDL principle for pattern mining
    3. Algorithms
        - Itemsets: Krimp (and SLIM)
        - Sequences: SQS
        - Graphs: Graph-MDL
    - Conclusion

Question

How can we find useful patterns?

Standard pattern mining

For a database db

- a pattern language L
- and a set of constraints C

the goal is to find the set of patterns P ⊆ L such that

- each p ∈ P satisfies each c ∈ C on db
- and P is maximal

That is, find all patterns that satisfy the constraints

Problems in pattern paradise

The pattern explosion problem

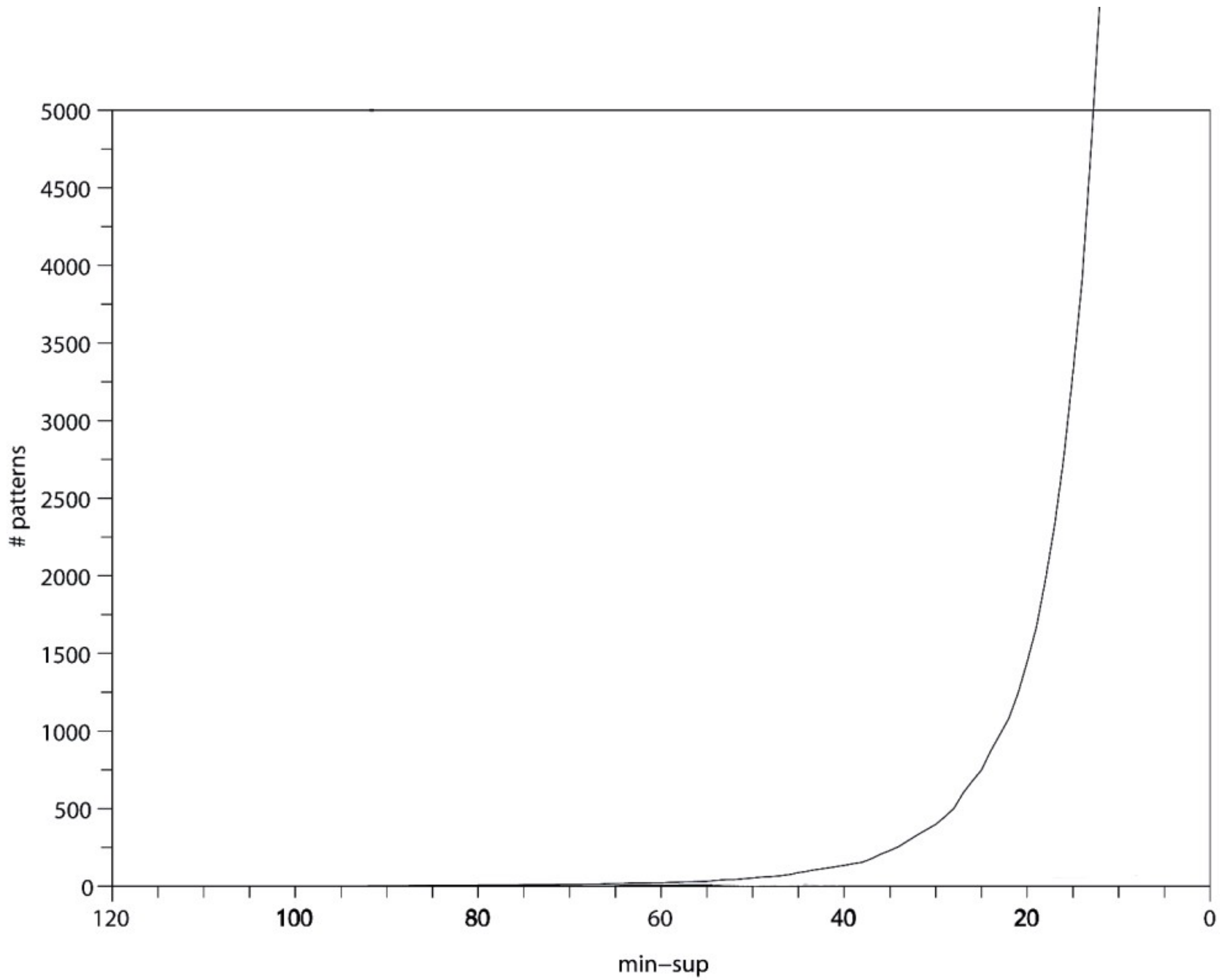
- ⇧ high thresholds
  ⇩ few, but <u>well-known</u> patterns

- ⇩ low thresholds
  ⇧ a <u>gazillion</u> patterns

Many patterns are redundant

Illustration: the *Wine* explosion

- **The *Wine* dataset has 178 rows, 14 columns**



Be careful what you wish for

The root of all evil is,

- we **ask** for **all** patterns
  that satisfy some constraints,

- while we want a **small** set
  that shows the structure of the data

What we ask for

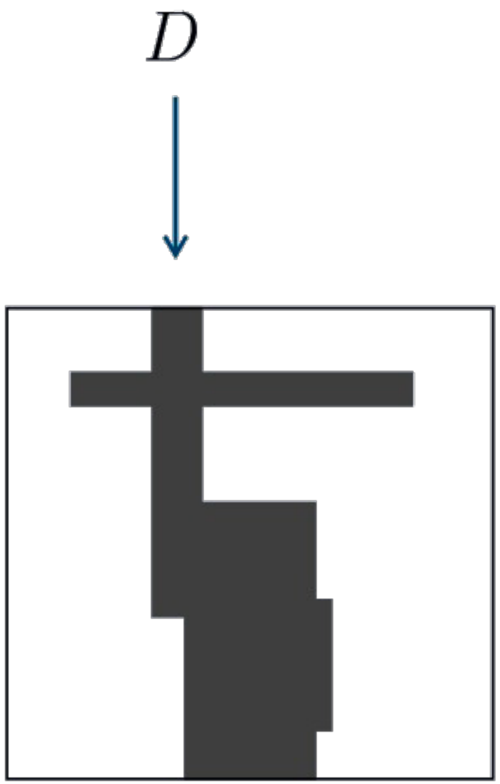
vs

What we really want

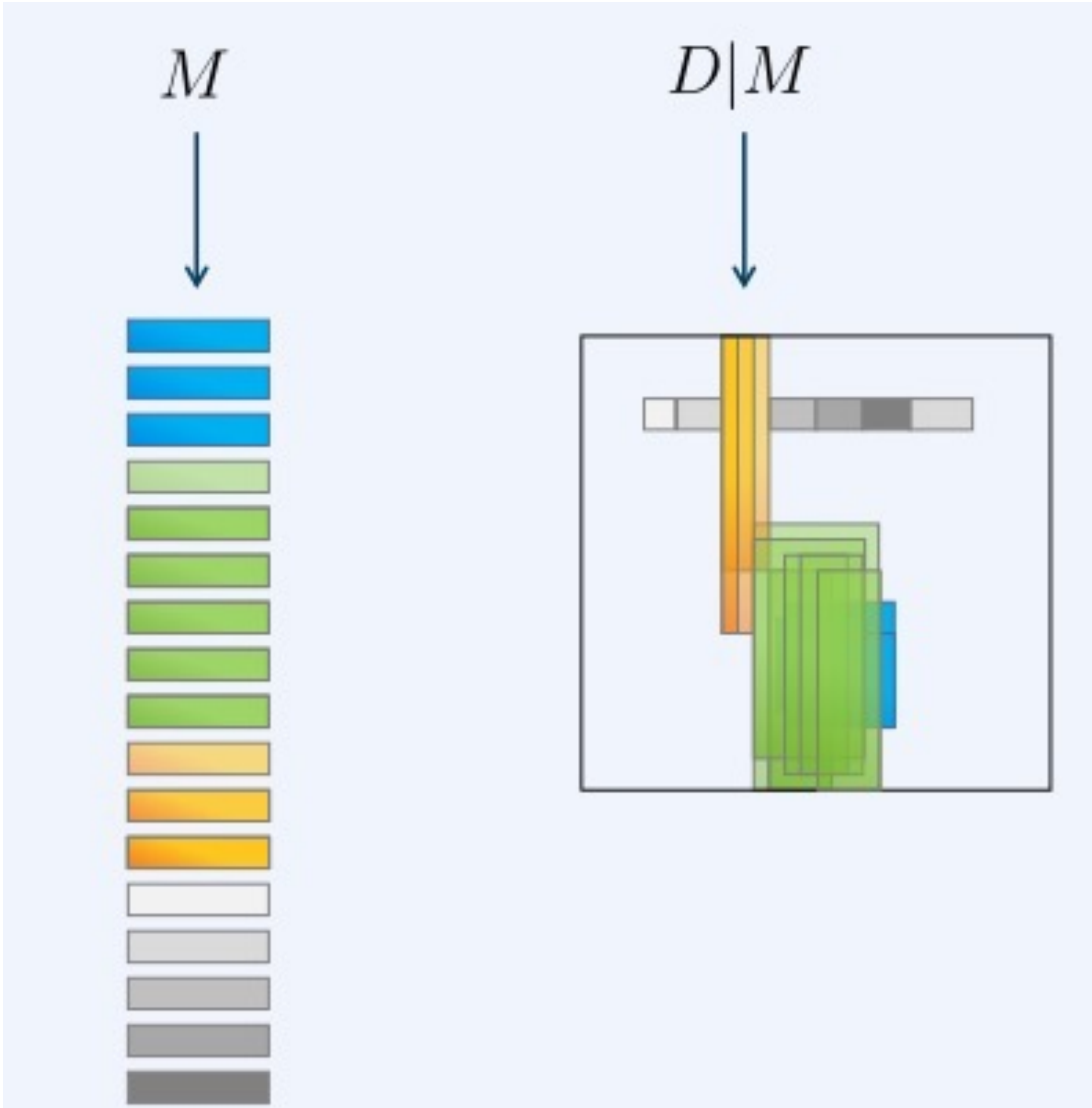
In other words, we should ask for a ***set of patterns*** **such that**

- all members of the set **satisfy** the constraints
- the set is **optimal** with regard to some criterion

What we ask for

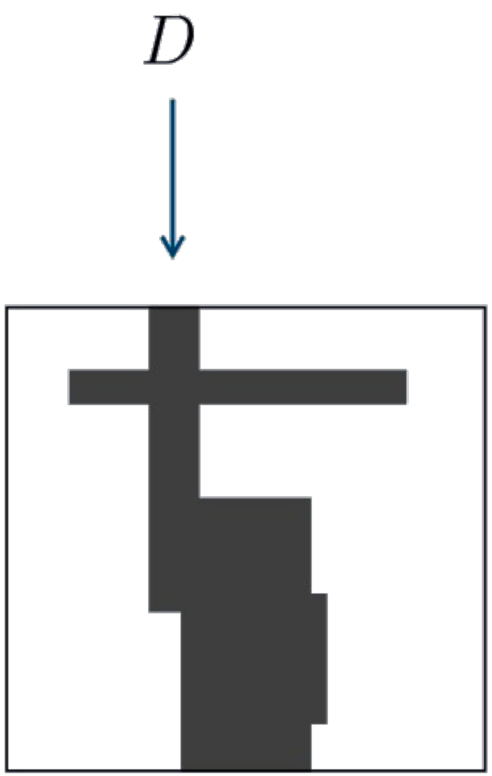


a toy 0-1 dataset

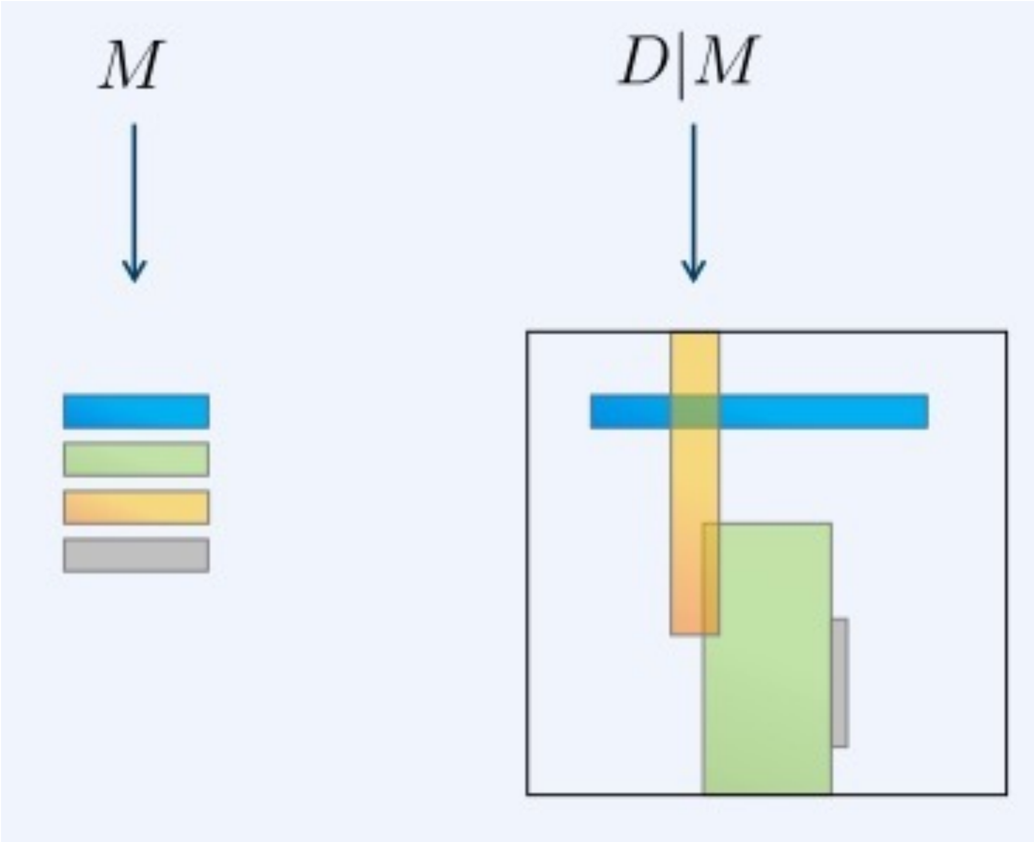


A pattern identifies local properties of the data (e.g., itemsets)

What we really want



a toy 0-1 dataset



How to do that?

How to find

- a subset of patterns
- describing data in a concise way

Outline

1. **Introduction**
2. **Compression based approaches**
   1. Motivations
   2. Information theory and the MDL principle
      - Information theory
      - The MDL principle for pattern mining
   3. Algorithms
      - Itemsets: Krimp (and SLIM)
      - Sequences: SQS
      - Graphs: Graph-MDL
   - Conclusion

Outline

1. **Introduction**
2. **Compression based approaches**
   1. Motivations
   2. Information theory and the MDL principle
      - **Information theory**
      - The MDL principle for pattern mining
   3. Algorithms
      - Itemsets: Krimp (and SLIM)
      - Sequences: SQS
      - Graphs: Graph-MDL
   - Conclusion

The Minimum Description Length (MDL) Principle

[Risanen'78, Grünwald'00]

Models describe the data

- that is, they capture regularities
- hence, in an abstract way, they compress it

The MDL principle makes this observation concrete:

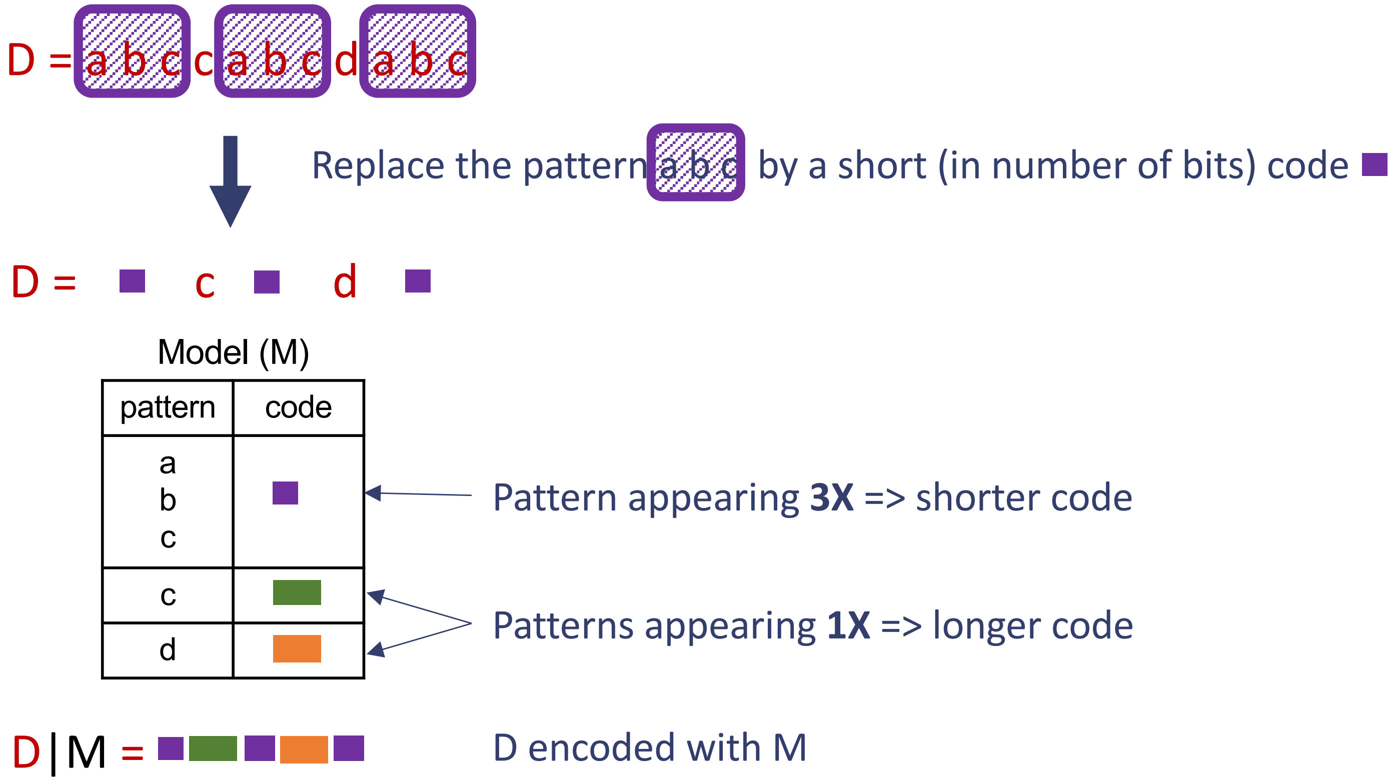
> **``The model that best describes the data is the one that best compresses the data.''**

The Minimum Description Length (MDL) Principle

[Risanen'78, Grünwald'00]

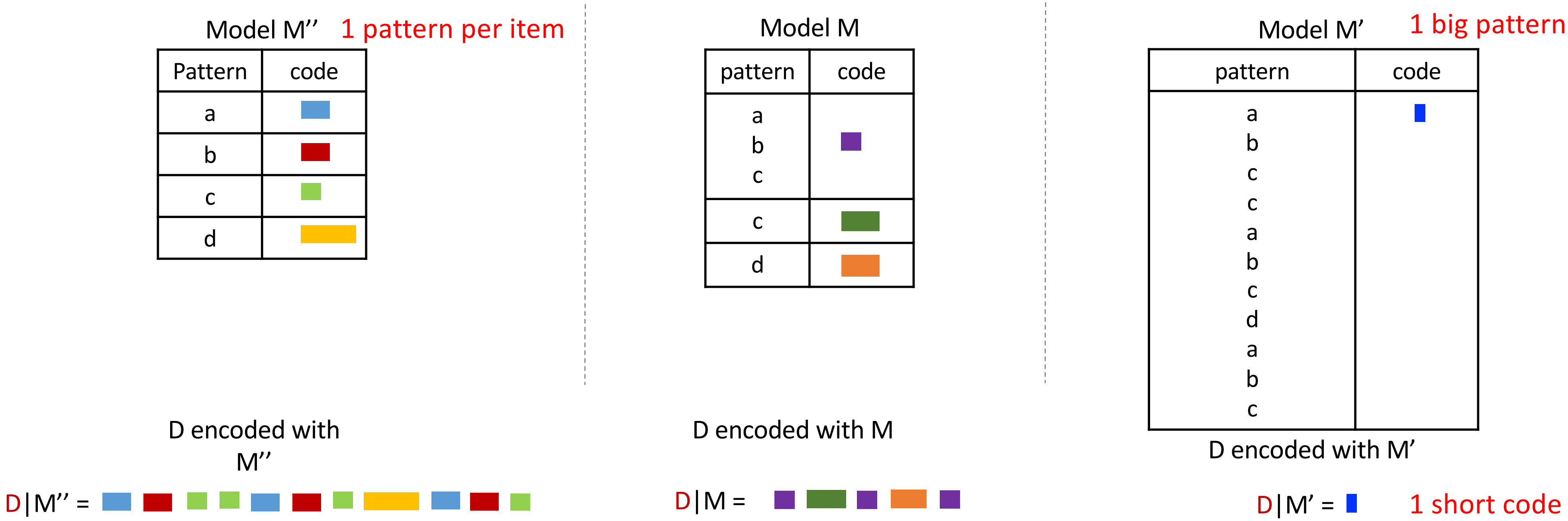
How to compress data?

⇒ take advantage of repetitions (patterns)



The Minimum Description Length (MDL) Principle
[Risanen'78, Grünwald'00]

How to find the BEST compression?
⇒ trade-off



The Minimum Description Length (MDL) Principle

[Risanen'78, Grünwald'00]

How to find the BEST compression?

⇒ trade-off

Principle: Find a trade-off by minimizing

L(D,M) = L(M) + L(D|M)

Description length of the model

Description length of the data encoded with the model

Information theory and compression

The MDL principle is related to Kolmogorov Complexity

Kolmogorov Complexity [Kolmogorov 1963]

the complexity of a string is the length of the smallest program that generates the string, and then halts

Kolmogorov Complexity is the ultimate compression

- recognizes and exploits **any** structure
- Uncomputable however…

Outline

1. **Introduction**
2. **Compression based approaches**
   1. Motivations
   2. Information theory and the MDL principle
      - Information theory
      - **The MDL principle for pattern mining**
   3. Algorithms
      - Itemsets: Krimp (and SLIM)
      - Sequences: SQS
      - Graphs: Graph-MDL
   - Conclusion

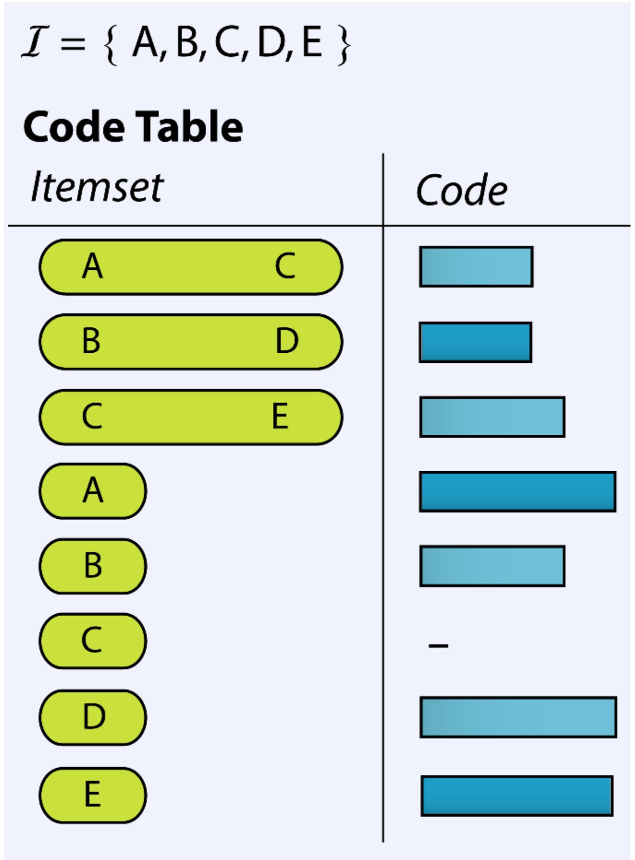
The MDL principle for pattern mining

- **Goal: reduce the number of extracted patterns**

- **Existing approaches for several pattern languages [Galbrun'22]**
  - Examples
    - Itemsets: KRIMP [Vreeken+'11], SLIM [Smets+'12]
    - Sequences: SQS [Tatti+'12]
    - Graphs: GraphMDL [Bariatti+'20], GraphMDL+ [Bariatti+'21]

Model = Code Table

- **Code Table (CT)**
  - Set of itemsets + encoding



Encoding a database: example

Code Table

Itemset	*Usage*
A C	0
B D	0
C E	0
A	0
B	0
C	0
D	0
E	0

Transaction *t*

B C E

Encoding a database: example

Code Table

Itemset	*Usage*
A C	0
B D	0
C E	0
A	0
B	0
C	0
D	0
E	0

Transaction *t*

B C E

Encoding a database: example

Code Table

Itemset	*Usage*
A C	0
B D	0
C E	0 + 1
A	0
B	0
C	0
D	0
E	0

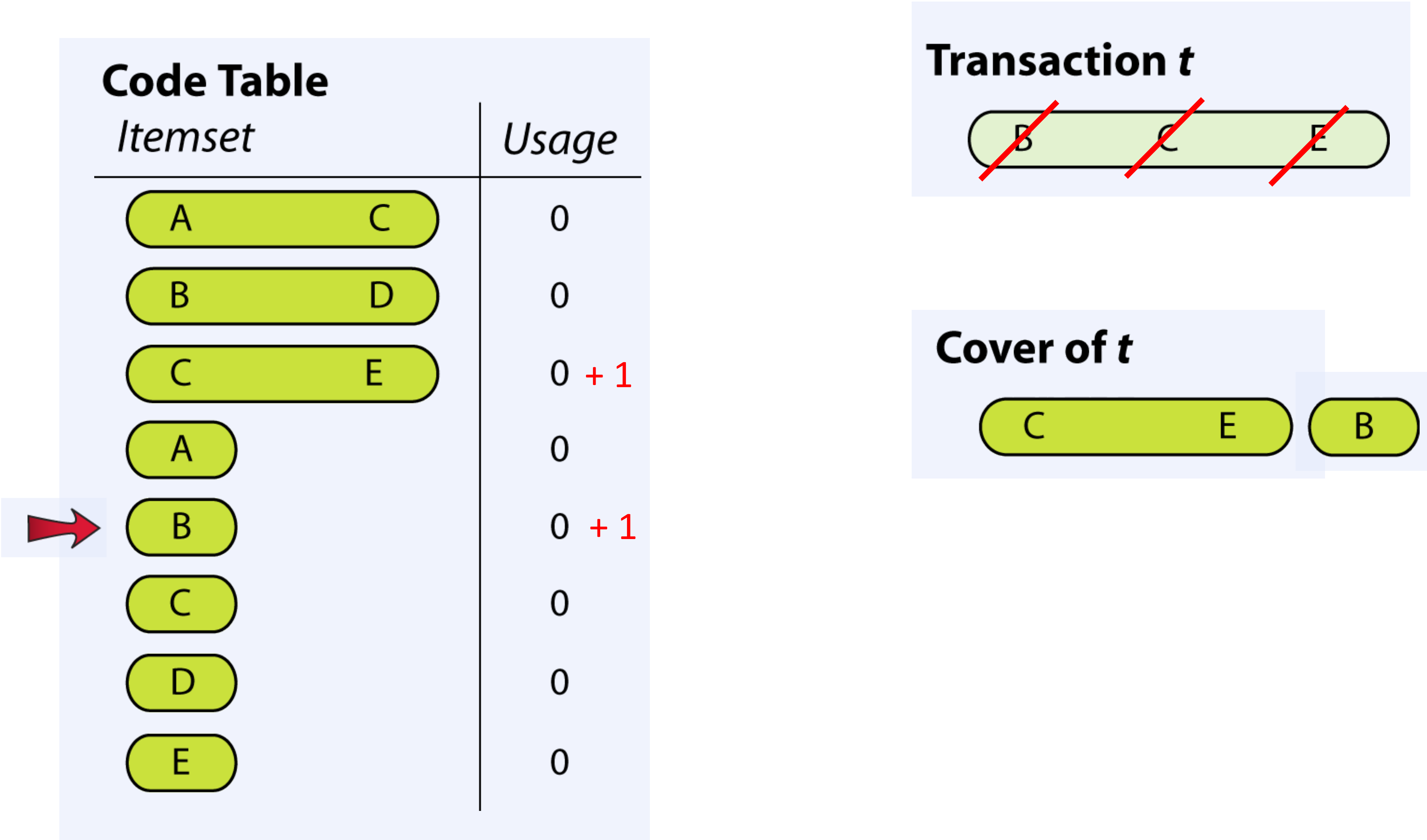
Transaction *t*

B ~~C~~ ~~E~~

Cover of *t*

C E

Encoding a database: example



Encoding a database: exercise

- **Exercise:**
  - Let us consider the following database and code table
  - Find the cover of the transactions
  - and the updated usages of the code table

Database

A	C	E	
B			
A	C		
A	B	C	D
B	C	E	
D			
B	C	D	E
A	B	D	

Code Table

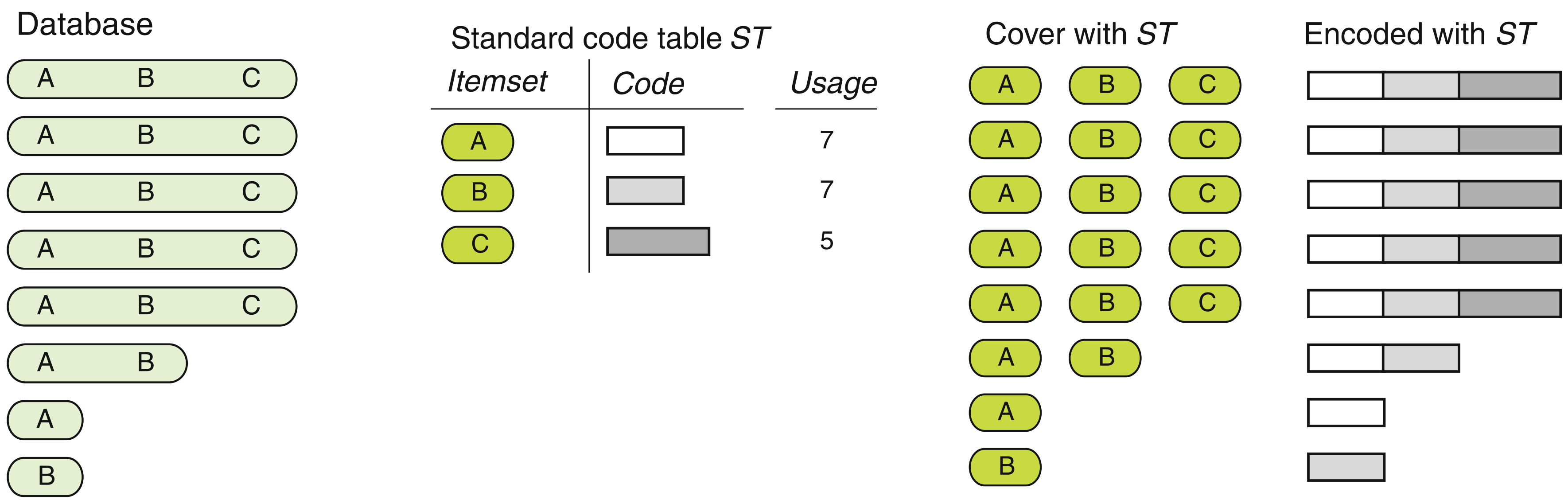
Itemset	*Usage*
A C	0
B D	0
C E	0
A	0
B	0
C	0
D	0
E	0

Solution

Standard Code Table

- **ST: Standard Code table**
  - = only and all singleton itemsets

- **Example**

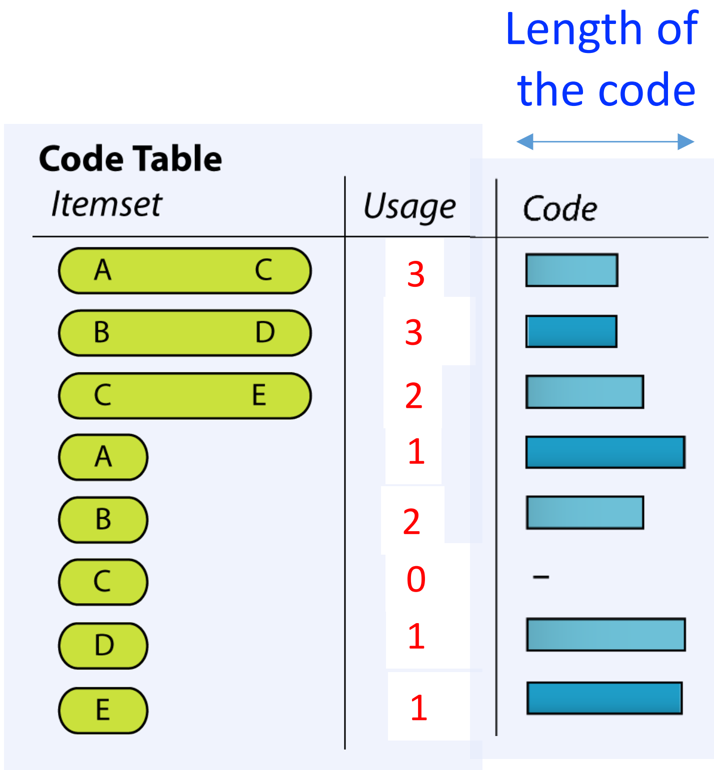


Example from paper [1]

Optimal codes [Shannon 1948, Thomas & Cover 1991]

- **What is the actual length of the codes? (in bits)**

- **For** $c \in CT$ **we define**
  - $P(c|CT) = \frac{usage(c)}{\sum_{i \in CT} usage(i)}$ where $usage(j) = |\{t \in D \mid j \; in \; cover(t, CT)\}|$
  - Example: P({A,C}) = 3/13



- **The optimal code for the coding distribution** *P* **assigns a code to** $c \in CT$ **with length:** [Shannon 1948]
  - $L(c|CT) = -\log P(c|CT)$
  - Example: L({A,C}) = -log(3/13) ≈ 2.12

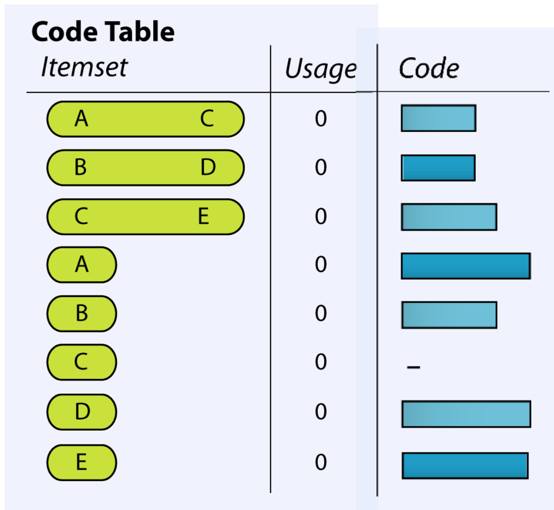
Encoding a code table

The size of a code table depends on

- the left column
  - length of itemsets as encoded with independence model
  - ST depends only from the data → used to measure CT size
- the right column
  - the optimal code length based on usage of code table elements

Thus, the size of a code table is

$$L(CT \mid D) = \sum_{c \in CT : usage(c) \neq 0} L(c|ST) + L(c|CT)$$



Encoding a database

For *t* ∈ *D* we have

$$L(t|CT) = \sum_{c \in cover(t,CT)} L(c|CT)$$

Description length (DL) for one transaction of D

Hence we have

$$L(D|CT) = \sum_{t \in D} L(t|CT)$$

Sum of the DL of all transactions of D

The Total Size

The total size of data and code table is

$$L(D, CT) = L(CT|D) + L(D|CT)$$

Description length of the model

Description length of the data encoded with the model

- **This is the MDL measure that we want to minimize**

And now, the optimal code table…

- **Easier said than done**
  - The number of possible code tables is huge
  - No useful structure to exploit

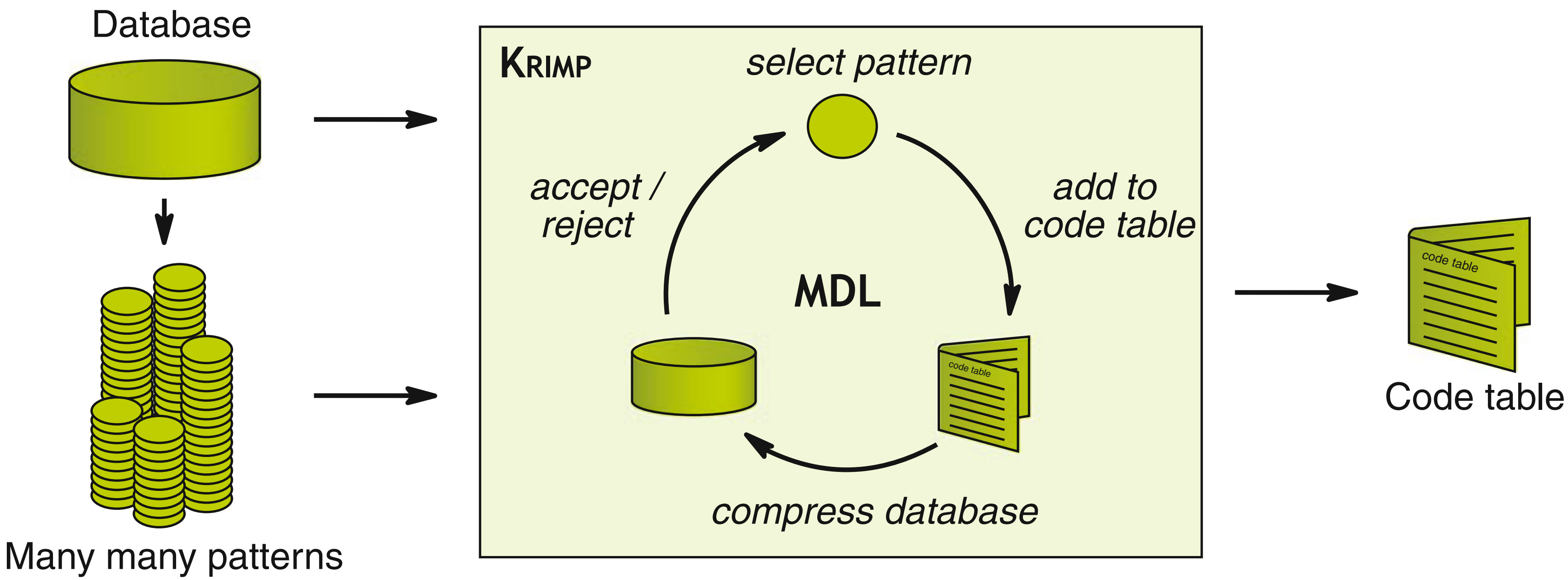
- **Hence, we resort to heuristics**

Outline

1. **Introduction**
2. **Compression based approaches**
   1. Motivations
   2. Information theory and the MDL principle
      - Information theory
      - The MDL principle for pattern mining
   3. Algorithms
      - Itemsets: **Krimp** (and SLIM)
      - Sequences: SQS
      - Graphs: GraphMDL
- Conclusion

KRIMP

- **KRIMP**
  - Based on MDL
  - And heuristics



KRIMP algorithm

Algorithm 3 The KRIMP Algorithm

Input: A transaction database $\mathcal{D}$ and a candidate set $\mathcal{F}$, both over a set of items $\mathcal{I}$

Output: A heuristic solution to the Minimal Coding Set Problem, code table CT

$CT \leftarrow$ **Standard Code Table**($\mathcal{D}$)

$\mathcal{F}_o \leftarrow \mathcal{F}$ in **Standard Candidate Order** — $supp_D(X)$↓ $|X|$↓ Lexicographically↑

for all $F \in \mathcal{F}_o \setminus \mathcal{I}$ **do** — Add the candidate one by one

  $CT_c \leftarrow (CT \cup F)$ in **Standard Cover Order** — $|X|$↓ $supp_D(X)$↓ Lexicographically↑

  if $L(\mathcal{D}, CT_c) < L(\mathcal{D}, CT)$ **then**

    $CT \leftarrow CT_c$

  end if

end for

return CT

Improvement by Pruning

- **Example**
  - Let us consider 3 code tables

$$CT_1 = \{\{X_1, X_2\}, \{X_1\}, \{X_2\}, \{X_3\}\}$$
$$CT_2 = \{\{X_1, X_2, X_3\}, \{X_1, X_2\}, \{X_1\}, \{X_2\}, \{X_3\}\}$$
$$CT_3 = \{\{X_1, X_2, X_3\}, \{X_1\}, \{X_2\}, \{X_3\}\}$$

  - And assume that $sup(\{X_1,X_2,X_3\}) = sup(\{X_1,X_2\}) -1$
  - KRIMP will never consider CT_3
  - But it is possible that $L(D,CT_3) < L(D, CT_2)$
    - => Problem: CT_3 will not be considered

- **Solution: pruning in the code table that KRIMP is considering**

Improvement by Pruning

- **When pruning?**
  - When an itemset *F* is added to the code table
  - At line 6 of KRIMP Algorithm

- **Which itemsets have to be tested?**
  - Only ones for which the usage has decreased
  - Because their code length has increased

Algorithm 4 Code Table Post-Acceptance Pruning

Input: Code tables CT_c and CT, for a transaction database $\mathcal{D}$ over a set of items $\mathcal{I}$, where $\{X \in CT\} \subset \{Y \in CT_c\}$ and $L(\mathcal{D}, CT_c) < L(\mathcal{D}, CT)$.

Output: Pruned code table CT_p, such that $L(\mathcal{D}, CT_p) \leq L(\mathcal{D}, CT_c)$ and $CT_p \subseteq CT_c$.

$PruneSet \leftarrow \{X \in CT \mid usage_{CT_c}(X) < usage_{CT}(X)\}$ — Only itemsets with decreased usage

while $PruneSet \neq \emptyset$ **do**

  $PruneCand \leftarrow X \in PruneSet$ with lowest $usage_{CT_c}(X)$ — usage↓ = code length↑

  $PruneSet \leftarrow PruneSet \setminus PruneCand$

  $CT_p \leftarrow CT_c \setminus PruneCand$

  if $L(\mathcal{D}, CT_p) < L(\mathcal{D}, CT_c)$ **then** — Compare the length of D without the itemset

    $PruneSet \leftarrow PruneSet \cup \{X \in CT_p \mid usage_{CT_p}(X) < usage_{CT_c}(X)\}$

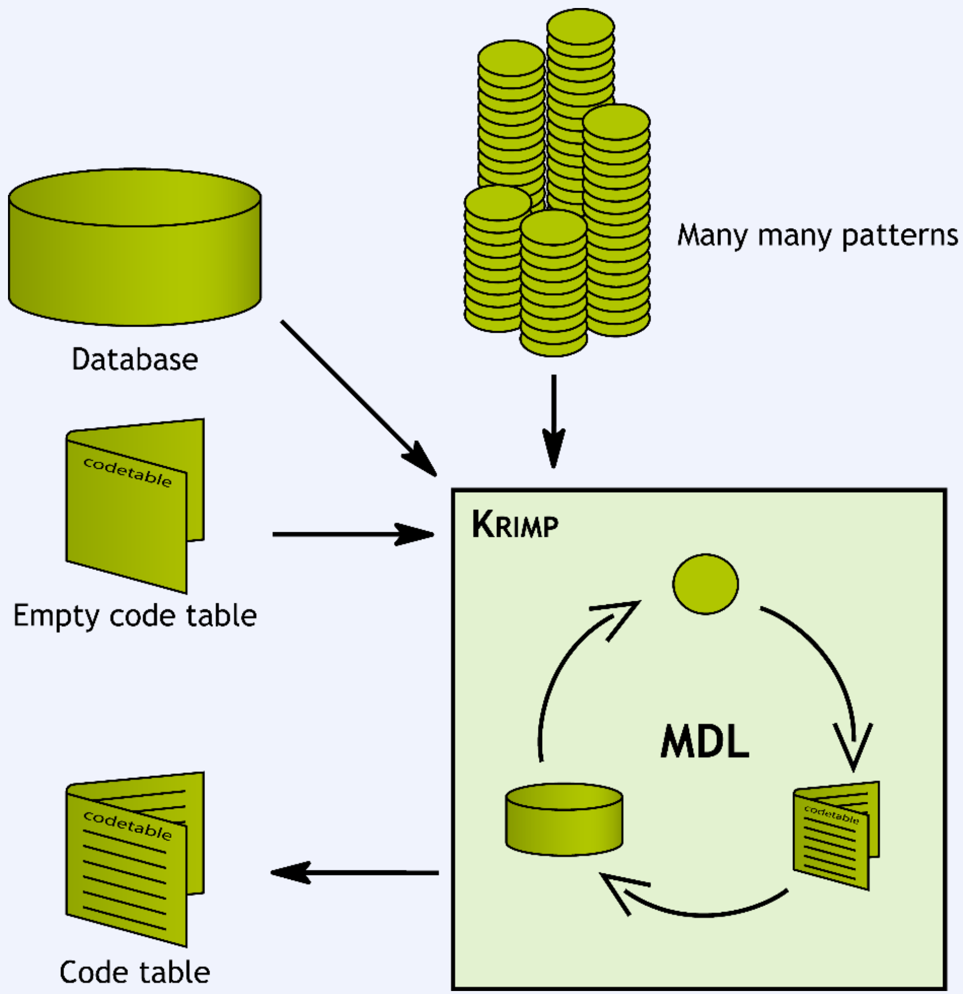
    $CT_c \leftarrow CT_p$

  end if

end while

return CT_c

KRIMP



- mine candidates from D
- iterate over candidates
  - Standard Candidate Order
- covers data greedily
  - no overlap
  - Standard Code Table Order
- select by MDL
  - better compression? candidates may stay, reconsider old elements

(Vreeken et al 2011)

So, are KRIMP code tables good?

At first glance, yes

- the code tables are characteristic in the MDL-sense
  - they compress well
- the code tables are small
  - they consist of few patterns
- the code tables are specific
  - they contain relatively long itemsets

But, are these patterns useful?

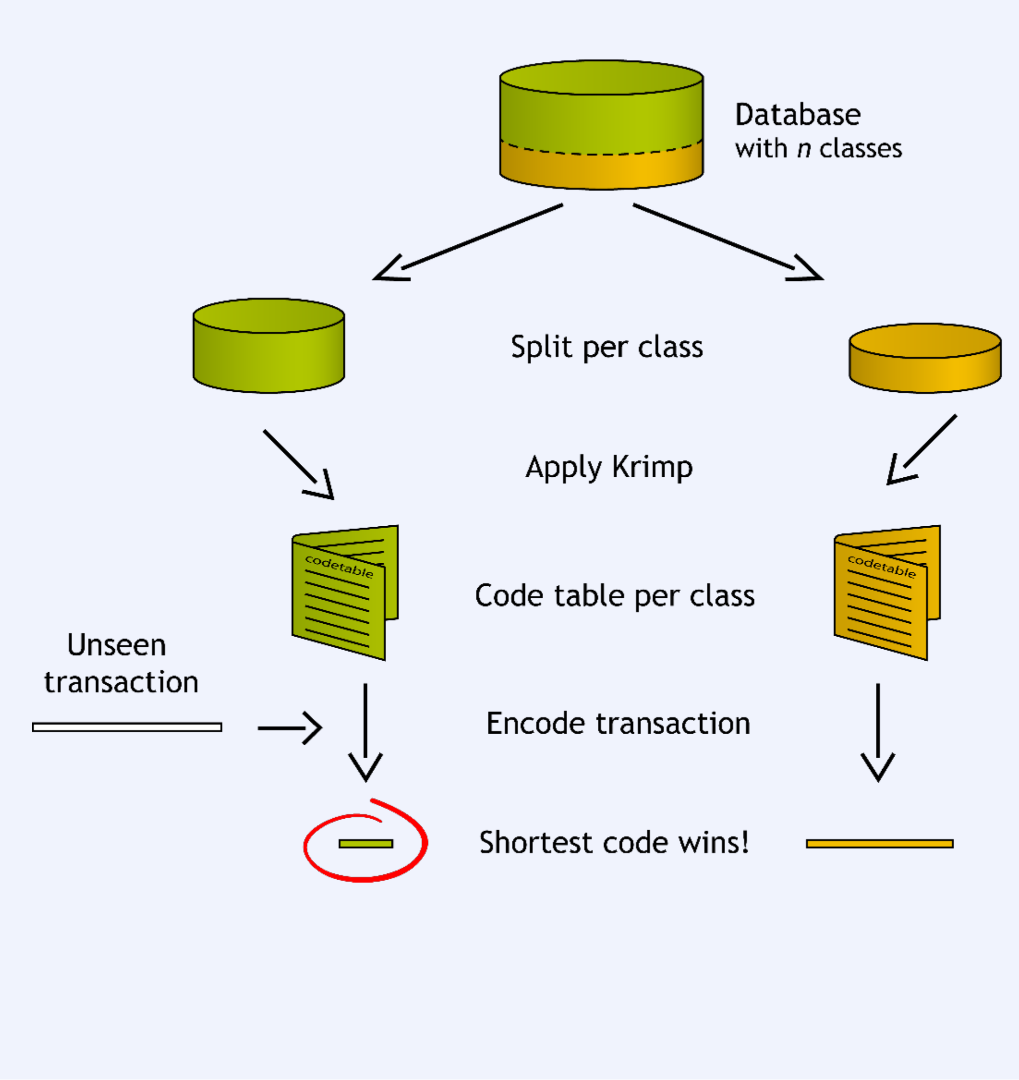
The proof of the pudding

The quality of the KRIMP code tables was tested by

- classification (ECML PKDD’06)
- measuring dissimilarity (KDD’07, ECML PKDD’15)
- generating data (ICDM’07)
- concept-drift detection (ECML PKDD’08)
- estimating missing values (ICDM’08)
- clustering (ECML PKDD’09)
- sub-space clustering (CIKM’09)
- anomaly detection (SDM’11, CIKM’12, SDM’17)
- characterising uncertain 0-1 data (SDM’11)
- tag-recommendation (IDA’12)
- Web semantic (Semantic Web’20)

Example: How to use KRIMP for Classification

- **The Krimp classifier**
  - Split database on class
  - Find code tables
  - Classify by compression

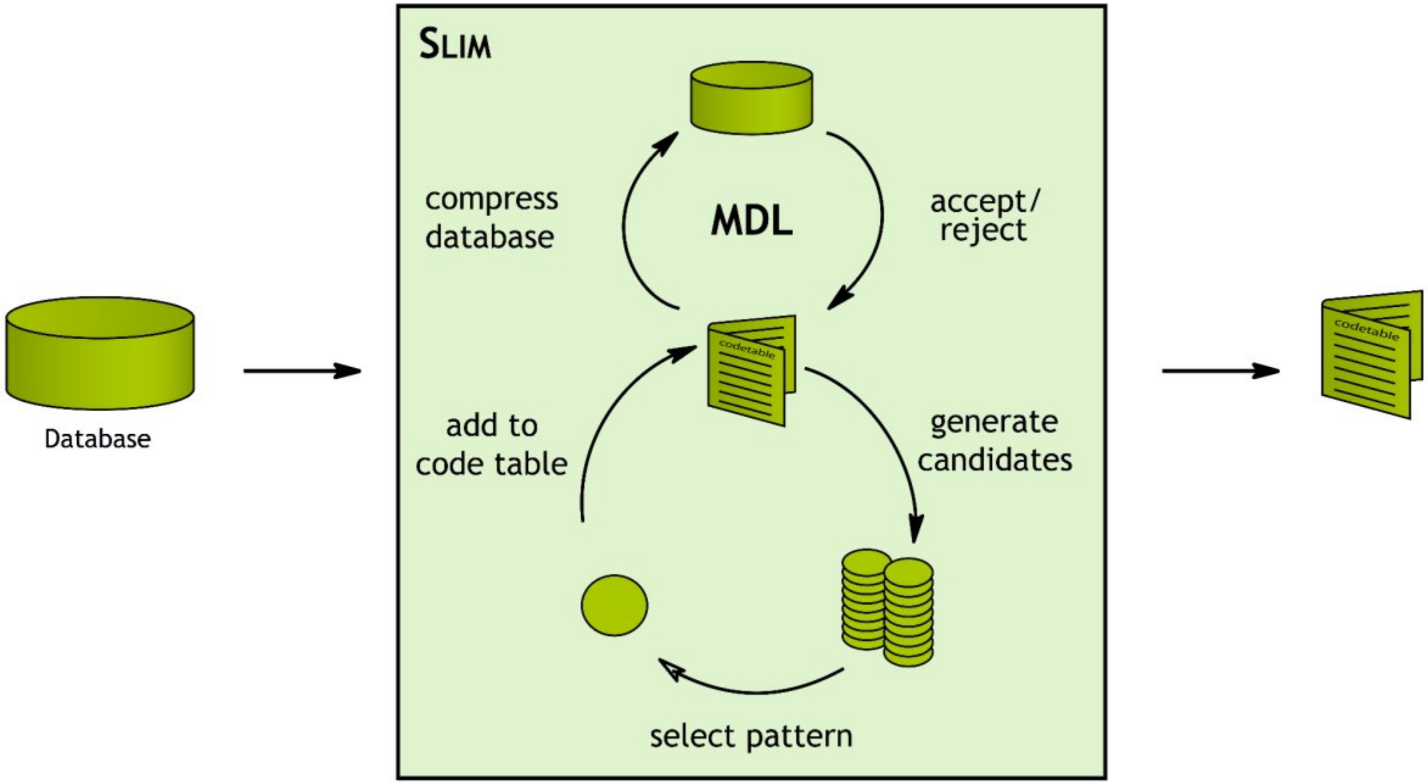


Outline

1. **Introduction**
2. **Compression based approaches**
   1. Motivations
   2. Information theory and the MDL principle
      - Information theory
      - The MDL principle for pattern mining
   3. Algorithms
      - Itemsets: Krimp (and **SLIM**)
      - Sequences: SQS
      - Graphs: GraphMDL
   - Conclusion

SLIM

- **Any-time version of KRIMP**
- **Generate-and-select instead of generate-then-select**



(Smets & Vreeken, 2012

SLIM algorithm

Algorithm 2 The SLIM Algorithm

Input: A transaction database $\mathcal{D}$ over a set of items $\mathcal{I}$

Output: A heuristic solution to the Minimal Coding Set Problem, code table CT

1. $CT \leftarrow$ **Standard Code Table**$(\mathcal{D})$
2. **for** $F \in \{X \cup Y : X, Y \in CT\}$ in **Gain Order do** usage↓
3. $\quad CT_c \leftarrow (CT \oplus F)$ in **Standard Cover Order** $|X|$↓ $supp_D(X)$↓ Lexicographically↑
4. $\quad$ **if** $L(\mathcal{D}, CT_c) < L(\mathcal{D}, CT)$ **then**
5. $\quad\quad CT \leftarrow$ *post-prune*(CT_c)
6. $\quad$ **end if**
7. **end for**
8. **return** CT

Note: At line 2 a list of the top-k candidates is kept in case the selected candidate fails the test at line 4

SLIM: example (with simplifications)

- **Let us assume**
  - A set of items I={A, B, C, D}
  - And a database *D*

transactions	description
T1	A B C
T2	B C D
T3	A B
T4	B C
T5	B D

- **Apply 2 steps of the loop of SLIM on** D
  - For a sake of simplicity
    - No postpruning

Memo
-log2(5/12) = 1.26
-log2(3/12) = 2
-log2(2/12) = 2.58
-log2(3/9)=1.58
-log2(2/9) = 2.16
-log2(1/9) = 3.16

Memo:

$$\text{Length}(t) = -\log 2\left(\frac{\text{usage}(t)}{\Sigma_{y \in CT}\, \text{usage}(y)}\right)$$

Solution (to complete)

- **1. Computation of the standard code table**

CT = ST =

pattern	support	usage	usage list	length	code

Solution (to complete)

- **2.1 Selection of a candidate F as the union of 2 elements of CT**

- **3.1 Computation of the new CTc**

Solution (to complete)

- **4.1 L(D,CTc) < L(D, CT) ?**
  - L(D | CTc) = Assume : L(CTc | D) = 18.26
  - L(D | CT) = Assume : L(CT | D) = 15.16

Solution (to complete)

- **2.2 Selection of a candidate F as the union of 2 elements of CT**

- **3.1 Computation of the new CTc**

Solution (to complete)

- **4.1 L(D,CTc) < L(D, CT) ?**
  - L(D | CTc) = assume L(CTc | D) = 25.74

SLIM

- **Characteristics**
  - Any-time algorithm
  - It considers at each step the refinement of the CT that provides most gain
    - Thanks to heuristics
  - Parameter-free (no minsup)

Outline

1. **Introduction**
2. **Compression based approaches**
   1. Motivations
   2. Information theory and the MDL principle
      - Information theory
      - The MDL principle for pattern mining
   3. Algorithms
      - Itemsets: Krimp (and SLIM)
      - Sequences: **SQS**
      - Graphs: GraphMDL
   - Conclusion

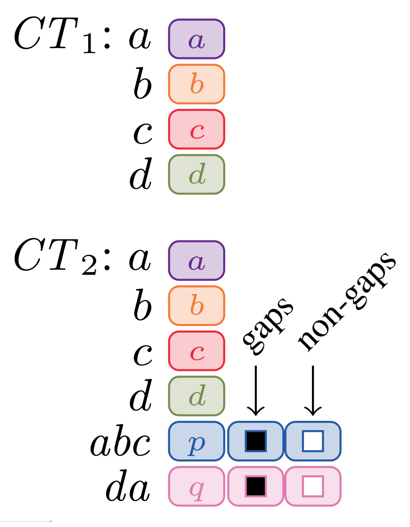
SQS

- SQS read « squeeze »
- MDL for sequences

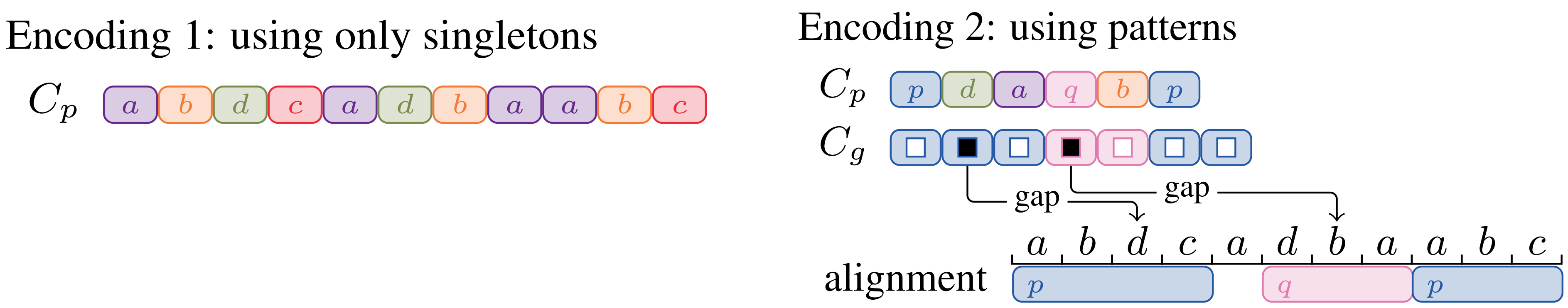
$$\text{Data } D: \; a \; b \; d \; c \; a \; d \; b \; a \; a \; b \; c$$

MDL for event sequences

- **Code table**
  - 4 columns
    - Pattern,
    - code,
    - gap code,
    - no gap code

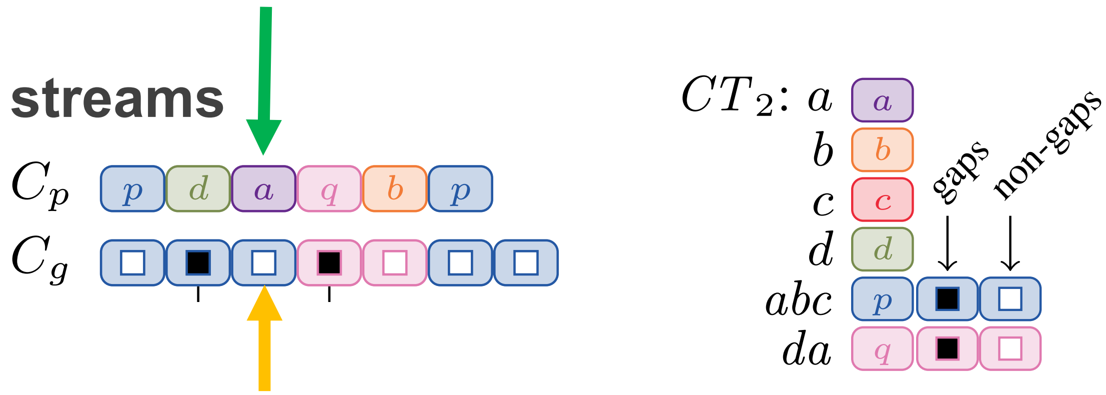


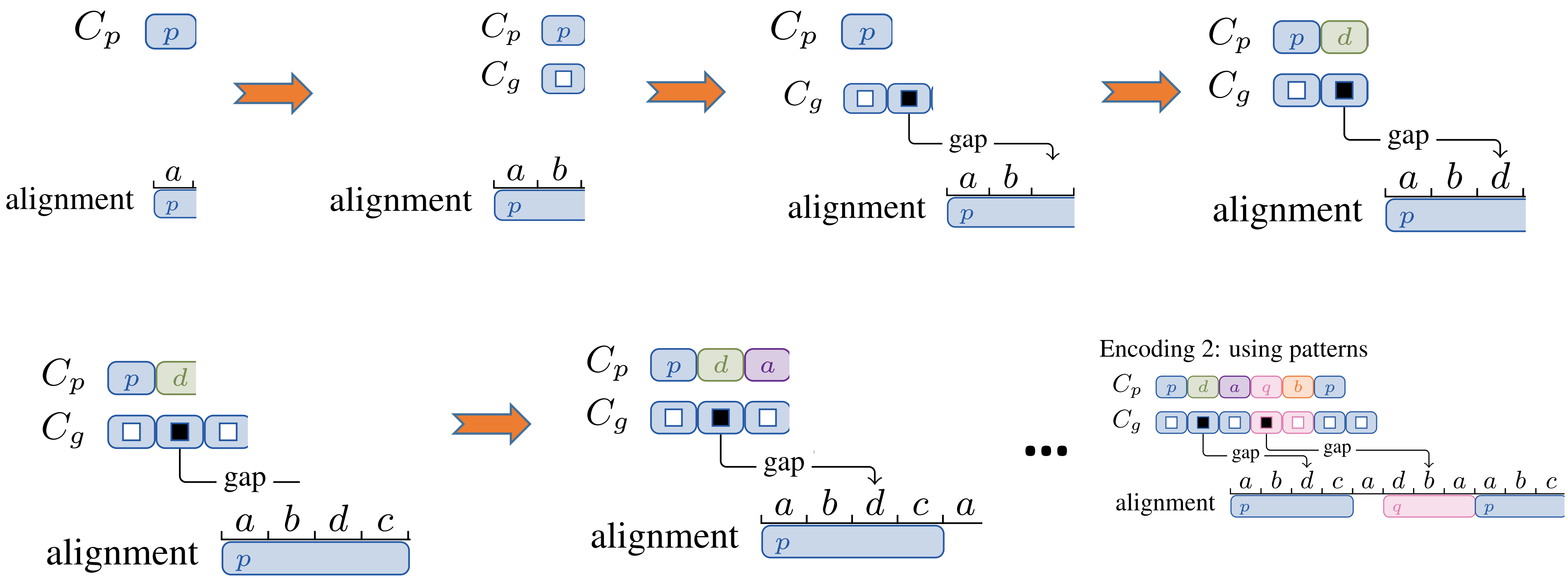
- **Encoded database = 2 streams**
  - Cp: pattern stream
  - Cg: gap stream



MDL for event sequences

- **Encoded database = 2 streams**
  - Cp: pattern stream
  - Cg: gap stream

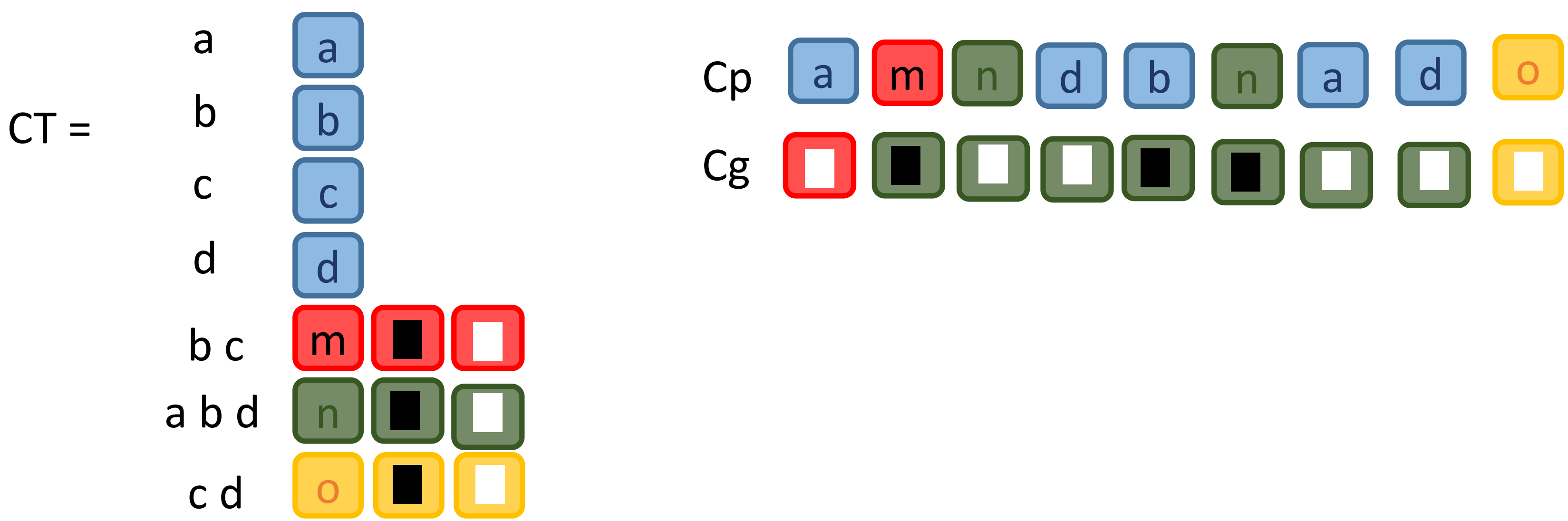




Exercise

- **Let us consider the following code table and the following streams**

- **Give the decoded sequence**



Encoding length

- Patterns

$$L(code_p(X) \mid CT) = -\log\left(\frac{usage(X)}{\sum_{Y \in CT} usage(Y)}\right)$$

- Let
  - *gaps*(*X*) = number of gaps in the usage of pattern X
  - *fills*(*X*) = number of non-gaps in the usage of pattern X
    - Note: $fills(X) = usage(X)(|X| - 1)$

- Gaps

$$L(code_g(X) \mid CT) = -\log\left(\frac{gaps(X)}{gaps(X) + fills(X)}\right)$$

- Non-gaps

$$L(code_n(X) \mid CT) = -\log\left(\frac{fills(X)}{gaps(X) + fills(X)}\right)$$

Encoding length

- Pattern stream

$$L(C_p \mid CT) = \sum_{X \in CT} usage(X) L(code_p(X))$$

- Gap stream

$$L(C_g \mid CT) = \sum_{\substack{X \in CT \\ |X|>1}} \Big(gaps(X) L(code_g(X)) + fills(X) L(code_n(X)) \Big)$$

- Database D with a code table CT

$$L(D \mid CT) = L_{\mathbb{N}}(|D|) + \sum_{S \in D} L_{\mathbb{N}}(|S|) + L(C_p \mid CT) + L(C_g \mid CT)$$

Number of sequences in D

Length of a sequence S in D

SQS (optional)

Just for information but not to learn: encoding of the code table

- Length of a pattern X

$$L(X, CT) = L_{\mathbb{N}}(|X|) + L_{\mathbb{N}}(gaps(X) + 1) + \sum_{x \in X} L(code_p(x \mid ST))$$

Length of the code table

$$L(CT \mid C) = L_{\mathbb{N}}(|\Omega|) + L_U(||D||, |\Omega|) + L_{\mathbb{N}}(|\mathcal{P}| + 1) + L_{\mathbb{N}}(usage(\mathcal{P}) + 1) + L_U(usage(\mathcal{P}), |\mathcal{P}|) + \sum_{X \in \mathcal{P}} L(X, CT)$$

SQS-CANDIDATES

- **Same idea as KRIMP**
  - Input:
    - a set of candidate patterns extracted by an other pattern mining algorithm
  - Method
    - Try to add one by one each candidate and see if the compression is better

Algorithm 3: $\text{SQS-CANDIDATES}(\mathcal{F}, D)$

input : candidate patterns $\mathcal{F}$

output : set of non-singleton patterns $\mathcal{P}$ that heuristically minimise the **Minimal Code Table Problem**

(Sort patterns) order patterns $X \in \mathcal{F}$ based on $L(D, \{X\})$;

$\mathcal{P} \leftarrow \emptyset$;

foreach $X \in \mathcal{F}$ in order **do**

(With pattern, gain?)   **if** $L(D, \mathcal{P} \cup X) < L(D, \mathcal{P})$ **then**

        $\mathcal{P} \leftarrow \text{PRUNE}(\mathcal{P} \cup X, D, \textbf{false})$; (Pruning step)

$\mathcal{P} \leftarrow \text{PRUNE}(\mathcal{P}, D, \textbf{true})$;

order patterns $X \in G$ by $L(D, \mathcal{P}) - L(D, \mathcal{P} \setminus X)$;

return $\mathcal{P}$;

SQS-SEARCH

- **Same idea as SLIM**
  - Any-time algorithm
  - Free-parameter

Algorithm 6: SQS-SEARCH(D)

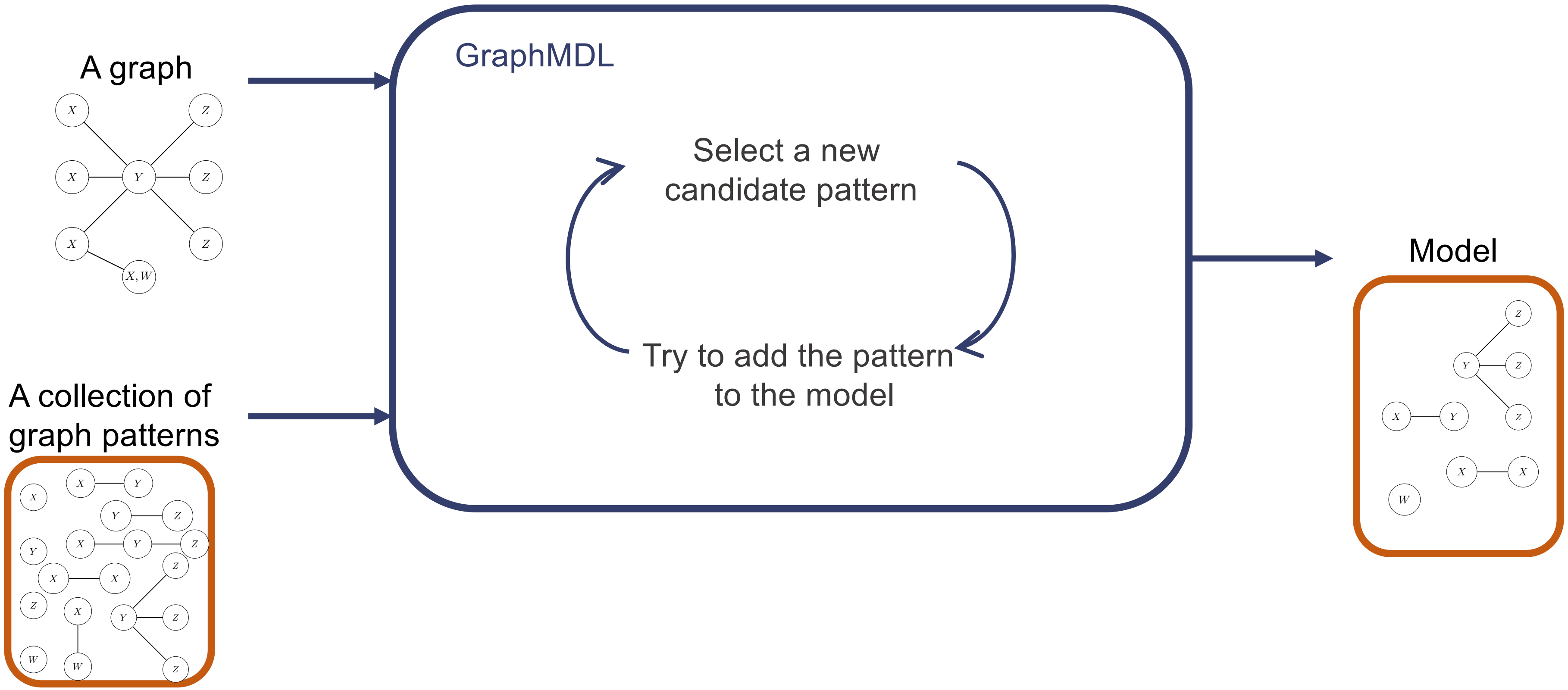
input : database D
output : significant patterns $\mathcal{P}$

$\mathcal{P} \leftarrow \emptyset$; $A \leftarrow$ SQS$(D, \emptyset)$;
while changes **do**
  $\mathcal{F} \leftarrow \emptyset$;
  foreach $P \in CT$ **do** add ESTIMATE(P, A, D) to $\mathcal{F}$;
  foreach $X \in \mathcal{F}$ ordered by the estimate **do**
    if $L(D, \mathcal{P} \cup X) < L(D, \mathcal{P})$ **then**
      $\mathcal{P} \leftarrow$ PRUNE$(\mathcal{P} \cup X, D,$ **false**$)$;
    if X is added **then** test recursively X augmented with events occurring in the gaps;
$\mathcal{P} \leftarrow$ PRUNE$(\mathcal{P}, D,$ **true**$)$;
order patterns $X \in G$ by $L(D, \mathcal{P}) - L(D, \mathcal{P} \setminus X)$;
return $\mathcal{P}$;

Outline

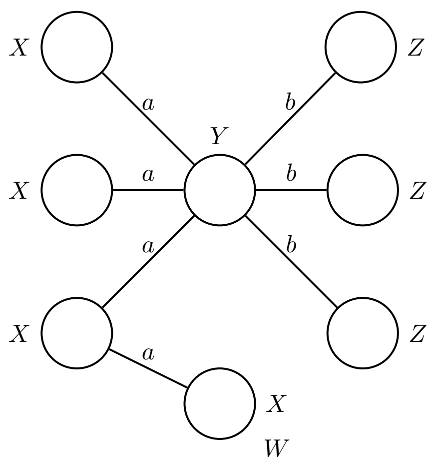
1. **Introduction**
2. **Compression based approaches**
   1. Motivations
   2. Information theory and the MDL principle
      - Information theory
      - The MDL principle for pattern mining
   3. Algorithms
      - Itemsets: Krimp (and SLIM)
      - Sequences: SQS
      - Graphs: **GraphMDL**
   - Conclusion

GraphMDL [IDA'20]



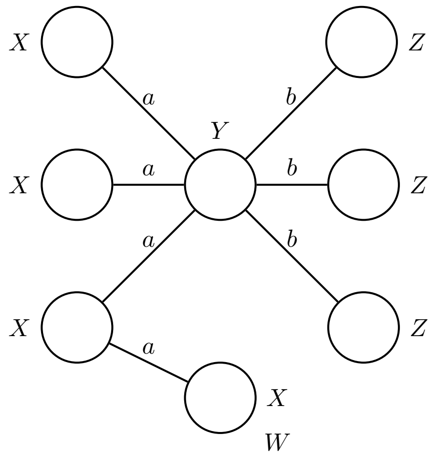
GraphMDL: Model

- **Graph**



GraphMDL: Model

- **Graph**

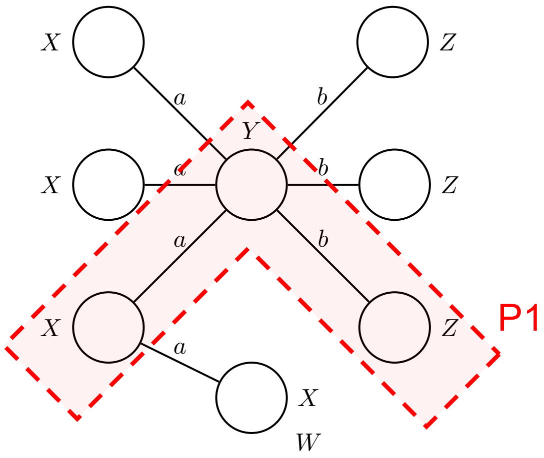


Model = Code table

	Pattern structure	Pattern code
P1	X — a — Y — b — Z (v1, v2, v3)	P 1
Pa	v1 — a — v2	Pa
Pw	W (v1)	Pw
Px	X (v1)	Px

GraphMDL: Model

- **Graph**



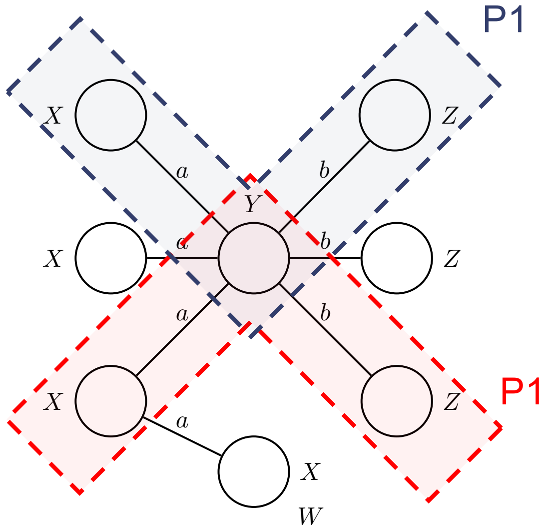
P1 

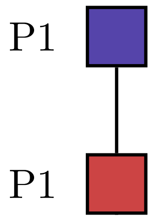
Model = Code table

	Pattern structure	Pattern code
P1	X Y Z; v1 —a— v2 —b— v3	P 1
Pa	v1 —a— v2	Pa
Pw	W; v1	Pw
Px	X; v1	Px

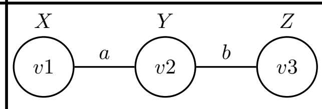
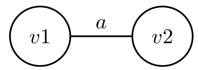
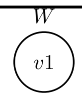
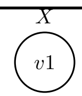
GraphMDL: Model

- **Graph**



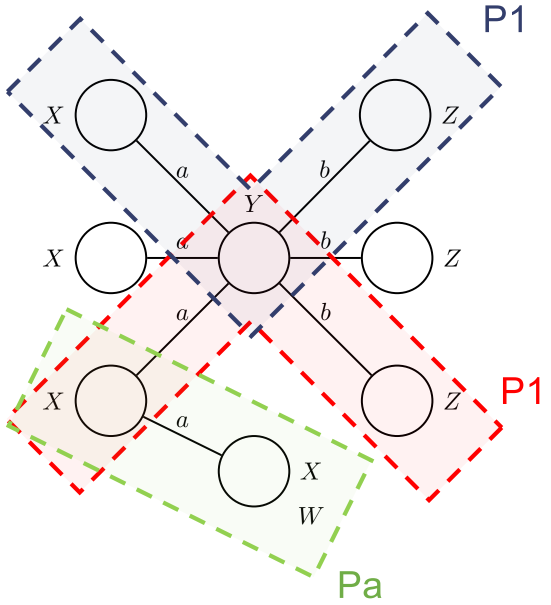


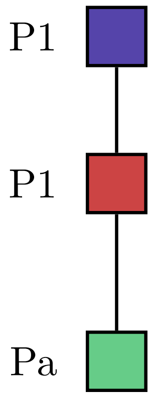
Model = Code table

	Pattern structure	Pattern code
P1		P 1
Pa		Pa
Pw		Pw
Px		Px

GraphMDL: Model

- **Graph**



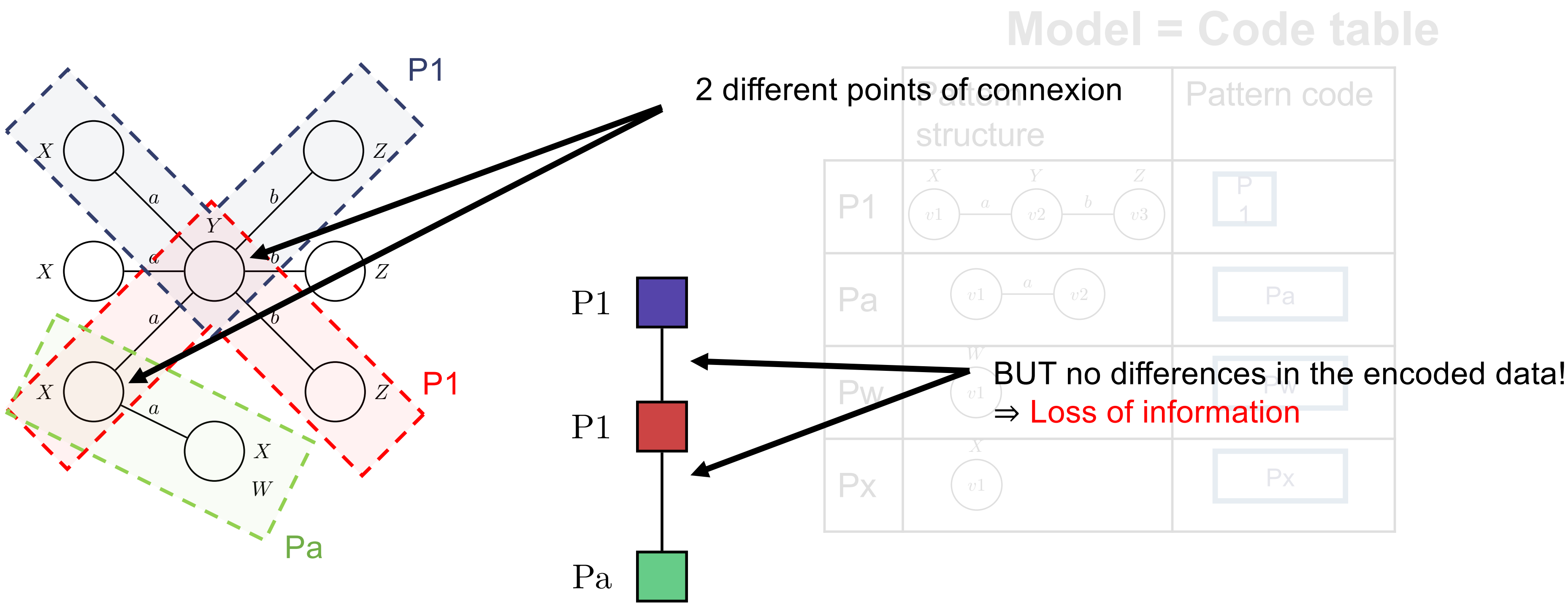


Model = Code table

	Pattern structure	Pattern code
P1	X, Y, Z: v1 —a— v2 —b— v3	P1
Pa	v1 —a— v2	Pa
Pw	W: v1	Pw
Px	X: v1	Px

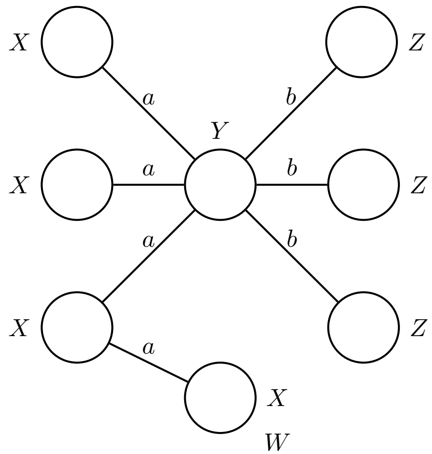
GraphMDL: Model

- **Graph**

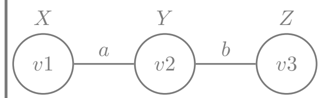
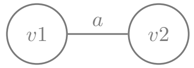
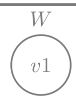
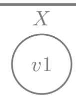


GraphMDL: Ports

- **Graph**



Model = Code table

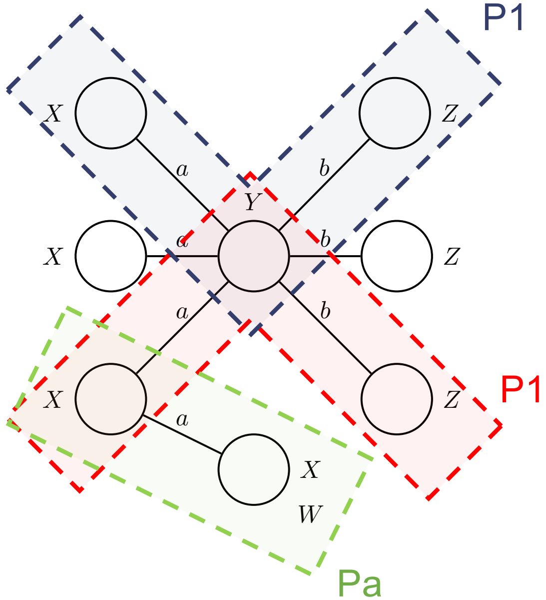
	Pattern structure	Pattern code	Port ID	Port code
P1		P1	v1 v2	v1 v2
Pa		Pa	v1 v2	v1 v2
Pw		Pw	v1	
Px		Px	v1	

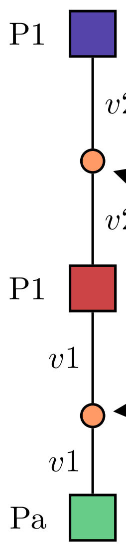
Ports

how pattern embeddings connect to each other to reconstruct the original graph

GraphMDL: Ports

- **Graph**





Model = Code table

	Pattern structure	Pattern code	Port ID	Port code
P1	X, Y, Z: v1 —a— v2 —b— v3	P1	v1 v2	v1 v2
Pa	v1 —a— v2	Pa	v1 v2	v1 v2
Pw	W: v1	Pw	v1	
Px	X: v1	Px	v1	

Ports

how pattern embeddings connect to each other to reconstruct the original graph

GraphMDL: Experiments

- **3 datasets**
  - AIDS-CA, AIDS-CM: about molecules (few labels, many cycles)
    - National Cancer Institute AIDS antiviral sceen data
  - UD-PUD-En: about dependency trees of english sentences (many labels, no cycles)
    - Universal dependencies project

- **Quantitative results**

Dataset	gSpan minsup	# Candidate patterns input	# Selected patterns output
AIDS-CA	20%	2194	115
AIDS-CA	15%	7867	123
AIDS-CA	10%	20596	148
AIDS-CM	20%	433	111
AIDS-CM	15%	779	131
AIDS-CM	10%	2054	163
AIDS-CM	5%	9943	225
UD-PUD-En	10%	164	162
UD-PUD-En	5%	458	249
UD-PUD-En	1%	6021	523
UD-PUD-En	0%	233434	773

- Drastic reduction
- Relatively stable in the number of selected patterns

GraphMDL+

- **GraphMDL+ as SLIM and SQS-Search**
  - Generate-and-select algorithm
  - Any-time algorithm
  - Free-parameter

Outline

1. **Introduction**
2. **Compression based approaches**
    1. Motivations
    2. Information theory and the MDL principle
        - Information theory
        - The MDL principle for pattern mining
    3. Algorithms
        - Itemsets: Krimp (and SLIM)
        - Sequences: SQS
        - Graphs: GraphMDL
    - Conclusion

If we have time
Mining Periodic Patterns with a MDL Criterion

Esther Galbrun, Peggy Cellier, Alexandre Termier, Bruno Crémilleux

Conclusions

- **MDL is great for picking important and useful patterns**

- **KRIMP, SQS, GRAPHMDL approximate the MDL ideal very well**
  - vast reduction of the number of itemsets
  - works for other pattern types equally well:
    itemsets, sequences, streams

- **Local patterns and information theory**
  - naturally induce good classifiers, clusterers, distance measures
  - with instant characterisation and explanation,
  - and, without (explicit) parameters

Conclusions

- **When using MDL for pattern mining**
  - Question 1: what kind of model to use?
    - Data modelisation
  - Question 2: how to encode the database? the model?
    - Information theory
  - Question 3: how to find the « best » (or at least one good) model?
    - Algorithmic

Outline

1. **Introduction**
2. **Compression based approaches**
   1. Motivations
   2. Information theory and the MDL principle
      - Information theory
      - The MDL principle for pattern mining
   3. Algorithms
      - Itemsets: Krimp (and SLIM)
      - Sequences: SQS
      - Graphs: Graph-MDL
   - Conclusion