A constraint-based pattern mining algorithm and its optimisation for multicore systems

Sofya Titarenko, Valeriy Titarenko, Georgios Aivaliotis, Jan Palczewski

EMiT 2019

What is pattern mining?

UNIVERSITY OF LEEDS

We can mine for interesting patterns, co-occurring patterns, **frequent patterns**, etc…

A frequent pattern is a set of events which are met often

Eg. datasets of transactions in supermarkets, road accidents, bioinformatics, environmental, health records, etc..

1 3 4

2 **3 5**

1 2 3 5 **1 2** met twice

2 5

1 2 3 5

Why do we mine for frequent patterns?

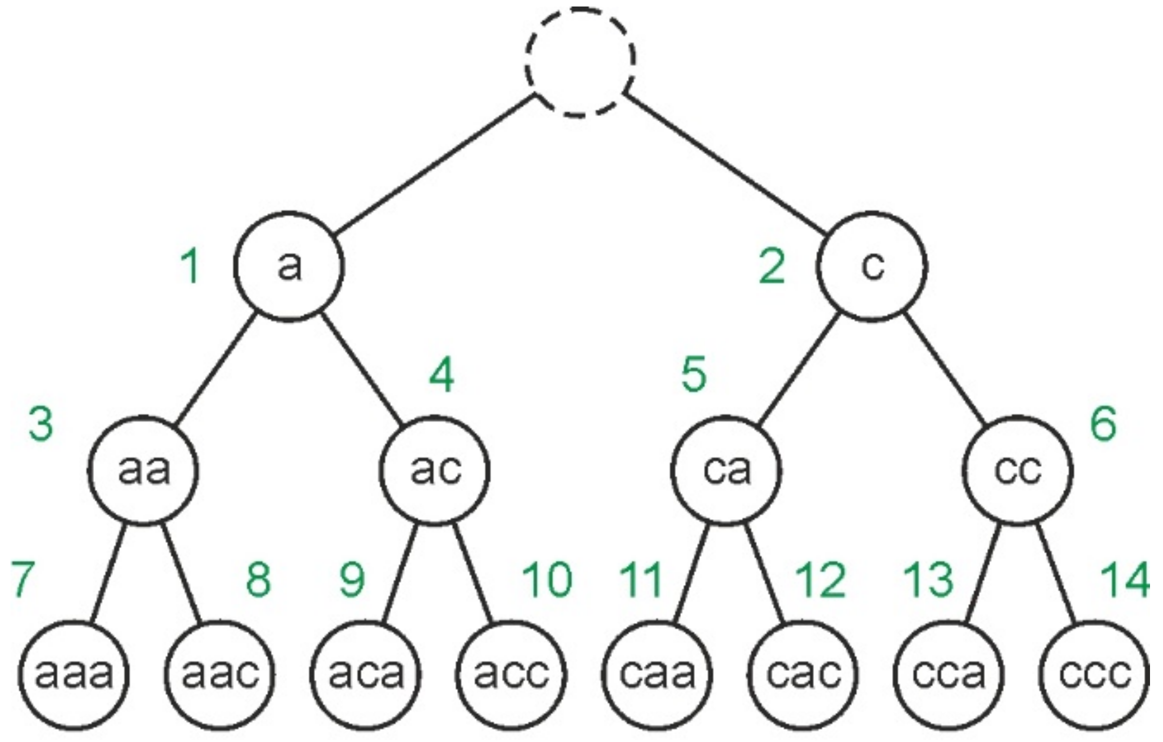
We often use found frequent patterns for the future analysis: clustering, building predictive models, classification, etc..

Challenges in frequent pattern mining

UNIVERSITY OF LEEDS

Storage Space

Computational time



Breadth first search

However…

We want to solve our problem in real-time. *Ex: making a medical decision*

We want to keep it running on stand-alone workstation. *Ex: working with sensitive datasets*

Challenges in frequent pattern mining

UNIVERSITY OF LEEDS

Taking time into account

Ex. Internet queries, medical monitoring, environmental monitoring etc..

Allowing for uncertainty in datasets

Human error, faults in sensors, sampling errors, etc..

More complexity, longer time, data storage challenge!

Additional constraints: item-based, temporal, etc..

Interested only in the patterns of a particular length

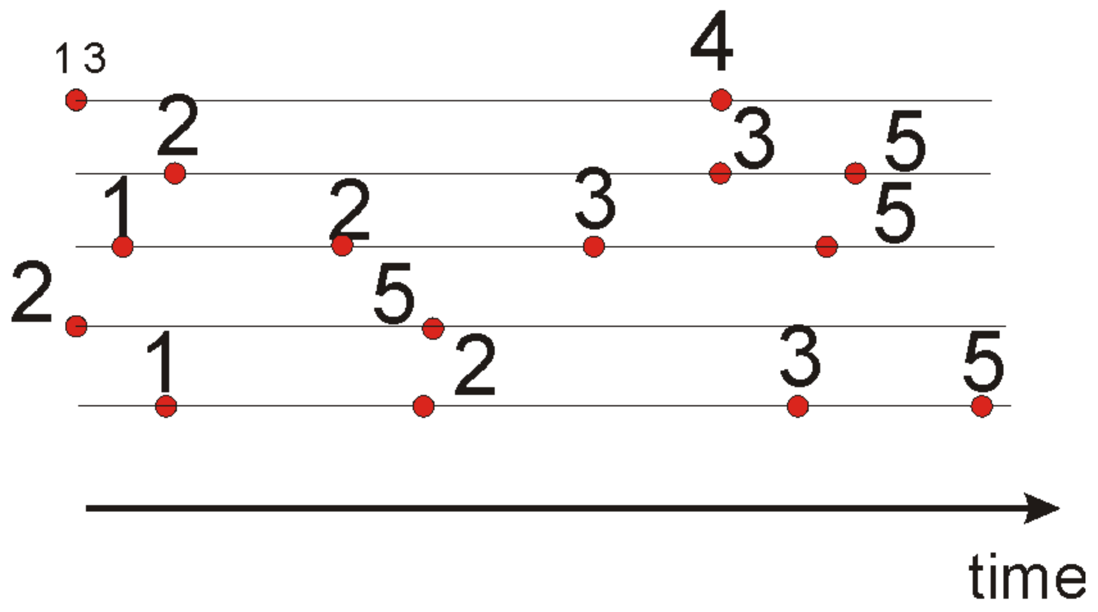
Patterns formed only from the events belonging to different categories

Challenges in frequent pattern mining

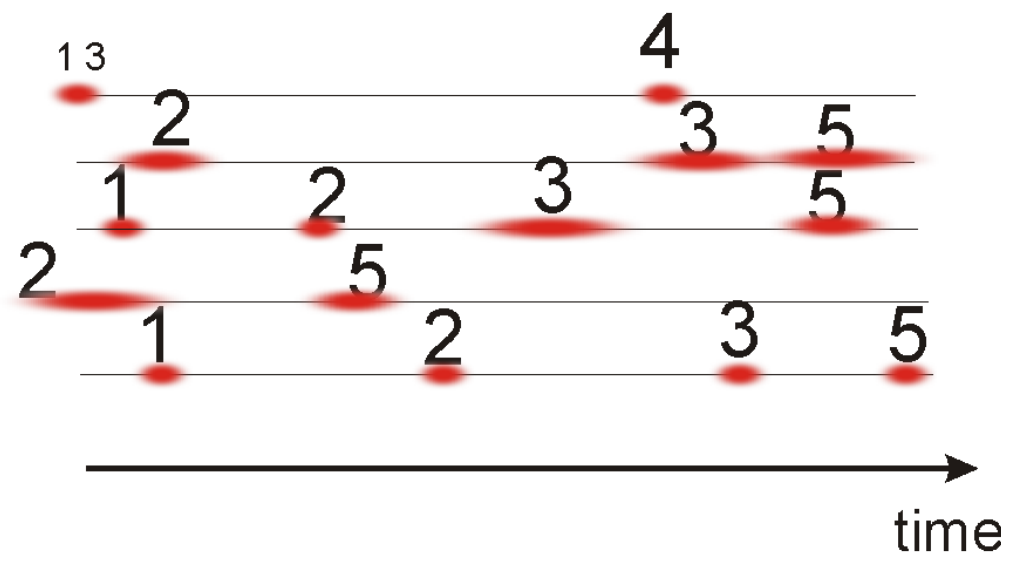
UNIVERSITY OF LEEDS

Time and uncertainty in time-points

Time-series



Adding uncertainty



Challenges in frequent pattern mining

UNIVERSITY OF LEEDS

Temporal and item-based constraints

Ex: medical records

1 4

2 3 5

1 2 **3 5**

2 5

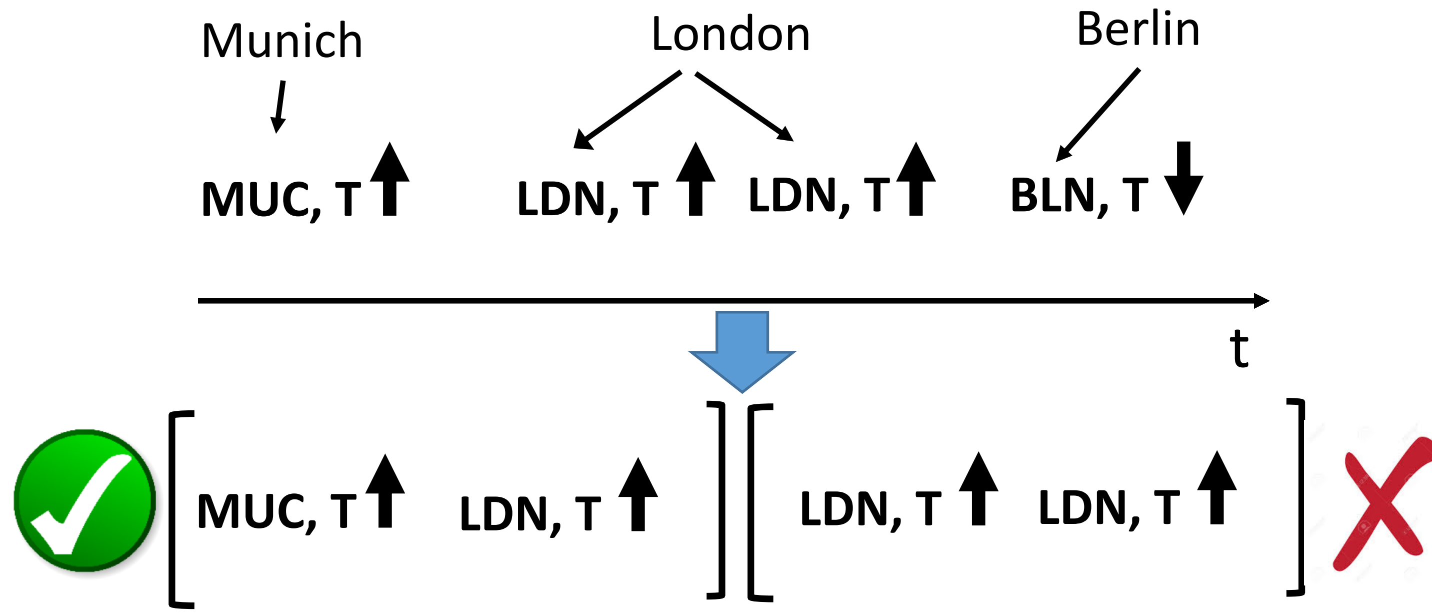
1 2 3 5

1 2 met twice

3 5 met once

Pattern is allowed to be no longer than a certain time period

Ex: weather dataset



*Pattern is allowed to contain **only items from different** groups*

Our algorithm FARPAM

UNIVERSITY OF LEEDS

Defines patterns to accommodate uncertainty, temporal and item-based constraints

Optimises calculations to be fast and efficient on a standalone multicore workstation

Steps for optimisation

Making storage more efficient

1. Clean dataset
2. “Pack” integers in a “smaller” storage space. Example: use 8 bit chars instead of 16/32/64 bit integers.
3. For binary vectors pack all the information in 32 bit.

Steps for optimisation

UNIVERSITY OF LEEDS

Improving processing time

1. Use of bitmap vectors and therefore binary logic operators (for ex. ADD)
2. Use multithreading and vectorisation wherever possible
3. Use caching memory strategy
4. Clean dataset
5. Store events in a “clever” way

FARPAM compared with FARPAMp

UNIVERSITY OF LEEDS

FARPAMp includes prior information:

- Specifically, the duration of uncertainty intervals is the same for all events of a certain type.
- The algorhithm could be modified for other prior constraints

Optimisation Results

UNIVERSITY OF LEEDS

Weather dataset, no uncertainty. Daily measurements over few years, 25 European cities

support	No. pat	apriori	apriori+opt	SPAM	FARPAM	FARPAMp
0.5	8332	120.1	18.7	14.7	1.19	1.21
0.4	46848	5942.1	51.9	50.4	3.66	3.87
0.3	157536	7519.8	**120.4**	**219.4**	**7.8**	**7.85**

Times are in seconds

~20 times faster than SPAM without uncertainty

810 records, 910,387 events

Optimisation Results

UNIVERSITY OF LEEDS

18,518 records, 304,719 events (Adult social care dataset, with uncertainty)

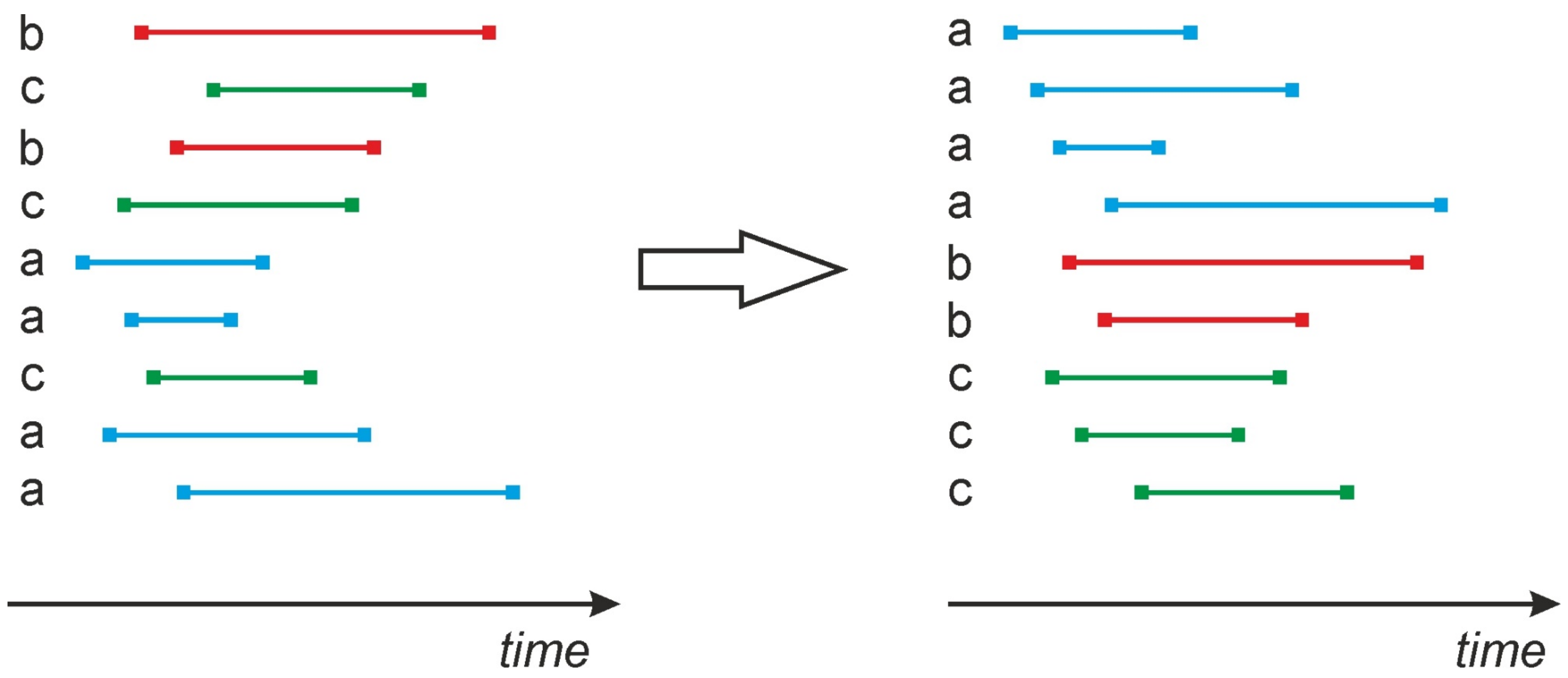
support	robustSpam	Apriori	Apriori+omp	FARPAM	FARPAMp
0.1	715.9	20.9	2.3	0.74	0.4
0.05	2370.7	91.7	8.1	1.06	1.05
0.03	5984.1	256.3	14.1	1.66	1.58
0.02	13045.1	602.2	26.8	2.3	1.88

~6000 times faster than robustSPAM with uncertainty

Only for pattern length [3,5]

Optimisation approaches

UNIVERSITY OF LEEDS



events	A B C B A A D B C A D
times	t1s, t1e t2s, t2e t3s, t3e…

events	A B C D
No. of unique events	4 3 2 2
times	t1s, t1e t2s, t2e t3s, t3e…

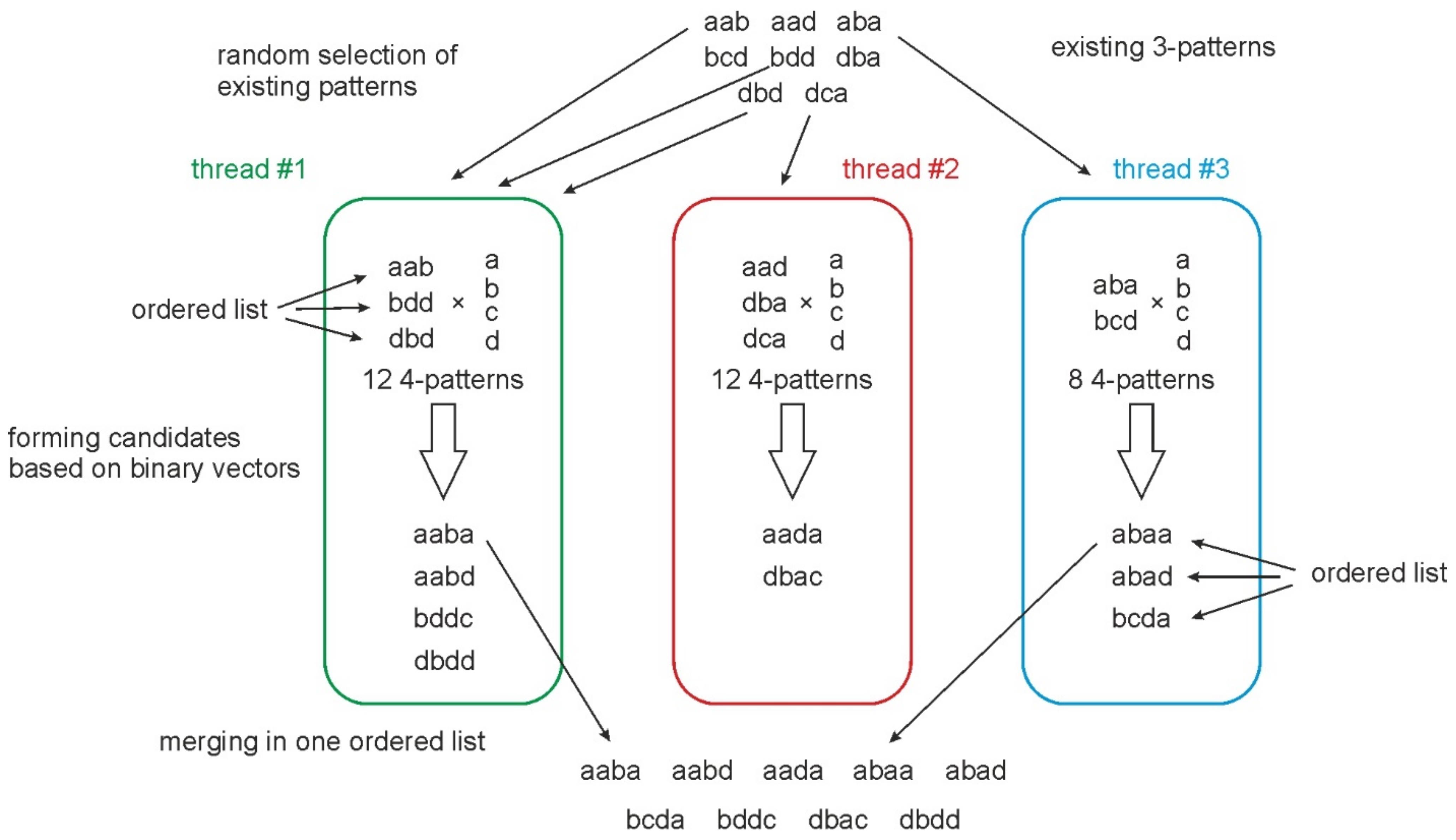
Optimisation approaches

	previously found patterns								candidate pattern
	bcd		acd		abd		abc		abcd
binary vectors	0	&	1	&	1	&	0	=	0
	1		0		1		1		0
	1		1		1		1		1
	1		1		1		1		1
	1		1		0		1		0
	1		1		1		1		1
	0		1		1		1		0
	1		1		1		1		1
	1		1		1		1		1
	1		0		1		1		0
	1		1		1		1		1
	1		1		1		0		0

OpenMP

Optimisation approaches

UNIVERSITY OF LEEDS



Conclusions

1. When working with Big Data it is very important to use storage space carefully;
2. Algorithmic and hardware optimisation allows reduction in storage space and considerably reduces calculative time;
3. Good algorithmic formulation allows flexibility in applications and makes possible future algorithmic modifications and extensions easier.
4. Use of prior knowledge can significantly speed up calculations

Challenges

UNIVERSITY OF LEEDS

1. Datasets

Problems:

1. Weather dataset (810 records, 910,387 events)
2. LCC (18,518 records, 304,719 events)

2. Constraints, uncertainty or additional information we want to keep….

Storage problem

Example:

1. suppose we found 2,000,000 of patterns with max length ~50. We need ~0.4 GB of memory
2. We want to accelerate problem using ID list approach. Suppose for problem 1. we have ~20,000 records. Then we need extra ~320GB

Time problem

If no uncertainty (weather example)

sup	N pat	apri ori	apri ori+ opt	SPA M	Alg1	Alg2
0.5	8332	120.1	18.7	14.7	1.19	1.21
0.4	46848	5942.1	51.9	50.4	3.66	3.87
0.3	157536	7519.8	**120.4**	**219.4**	**7.8**	**7.85**

Assuming that there's no coinciding events

+uncertainty (LCC example)

sup	robust Spam	Apriori	Apriori +omp	Alg1	Alg2
0.1	715.9	20.9	2.3	0.74	0.4
0.05	2370.7	91.7	8.1	1.06	1.05
0.03	5984.1	256.3	14.1	1.66	1.58
0.02	13045.1	602.2	26.8	2.3	1.88

Only for length [3,5]

Sofya Titarenko, Valeriy Titarenko, George Aivaliotis, Jan Palczewski, ***"Fast implementation of pattern mining algorithms with time stamp uncertainties and temporal constraints",*** to submit in journal of Big Data

The problem we solve

UNIVERSITY OF LEEDS

1. Reformulated pattern definition so to accommodate few types of pattern mining (itemset mining, time series mining, SPAM with sequences=1, time series with time uncertainty)

2. Cleaned dataset from the events which are not frequent

3. Store database in the following way: Dataset with only unique events for record, number of unique events, Dataset of times

4. Use of ID lists for frequent patterns. All ID lists are compressed in bitmap. When working with them we apply binary logic operators when it is possible

5. Check if pattern is frequent only if:

- All its subpatterns of length (n-1) are frequent
- Sup of the resultant logical multiplication of ID vectors is above min support value
- Check the corresponding entry only its binary value equals 1

6. Use the following property:
Suppose uncertainty interval is the same for alike events. If the interval starts earlier for the first event, then it also finishes earlier

7. Multithreading, vectorisation

The problem we solve

UNIVERSITY OF LEEDS

record	*events*	A B C B A A D B C A D
	times	t1s, t1e t2s, t2e t3s, t3e…

Classical way

3. Store database in the following way: Dataset with only unique events for record, number of unique events, Dataset of times

record	*events*	A B C D
	N of unique events	4 3 2 2
	times	t1s, t1e t2s, t2e t3s, t3e…

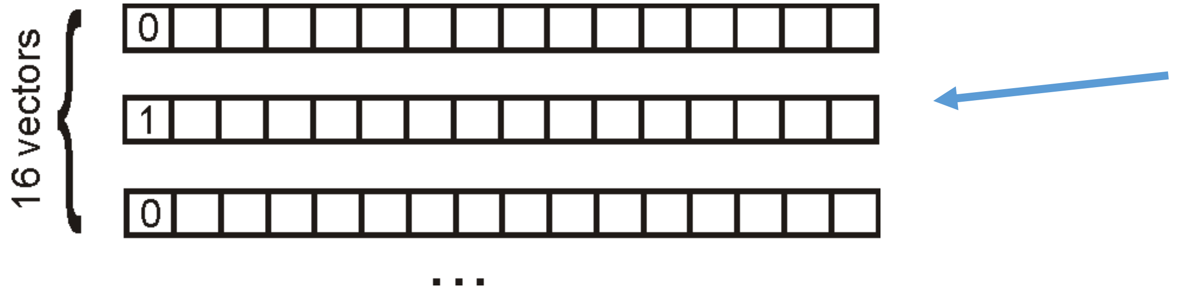
Proposed way

Advantages:

1. Dataset become more compact
2. Searching for pattern function works faster

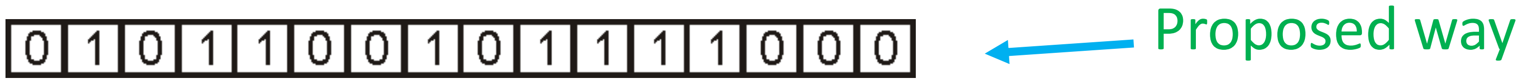
The problem we solve

UNIVERSITY OF LEEDS



16 vectors for **16** records!

4. Use of ID lists for frequent patterns. All ID lists are compressed in bitmap. When working with them we apply binary logic operators when it is possible



1 vector for **16** records!

UNIVERSITY OF LEEDS

If no uncertainty (weather example)

sup	N pat	apriori	apriori +opt	SPAM	Alg1	Alg2
0.5	8332	120.1	18.7	14.7	1.19	1.21
0.4	46848	5942.1	51.9	50.4	3.66	3.87
0.3	157536	7519.8	120.4	219.4	7.8	7.85

Assuming that there's no coinciding events

The proposed algorithms (Alg1 and Alg2) work up to **~30** times faster then open source code **SPAM** and **~7,000** times faster then previously developed **robustSPAM**

+uncertainty (LCC example)

sup	robust Spam	Apriori	Apriori +omp	Alg1	Alg2
0.1	715.9	20.9	2.3	0.74	0.4
0.05	2370.7	91.7	8.1	1.06	1.05
0.03	5984.1	256.3	14.1	1.66	1.58
0.02	13045.1	602.2	26.8	2.3	1.88

Only for length [3,5]