Constraint Programming for Itemset Mining with Multiple Minimum Supports

Mohamed-Bachir Belaid[1], Nadjib Lazaar[2]

30/11/2021

CAVIAR
Contraintes et Apprentissage

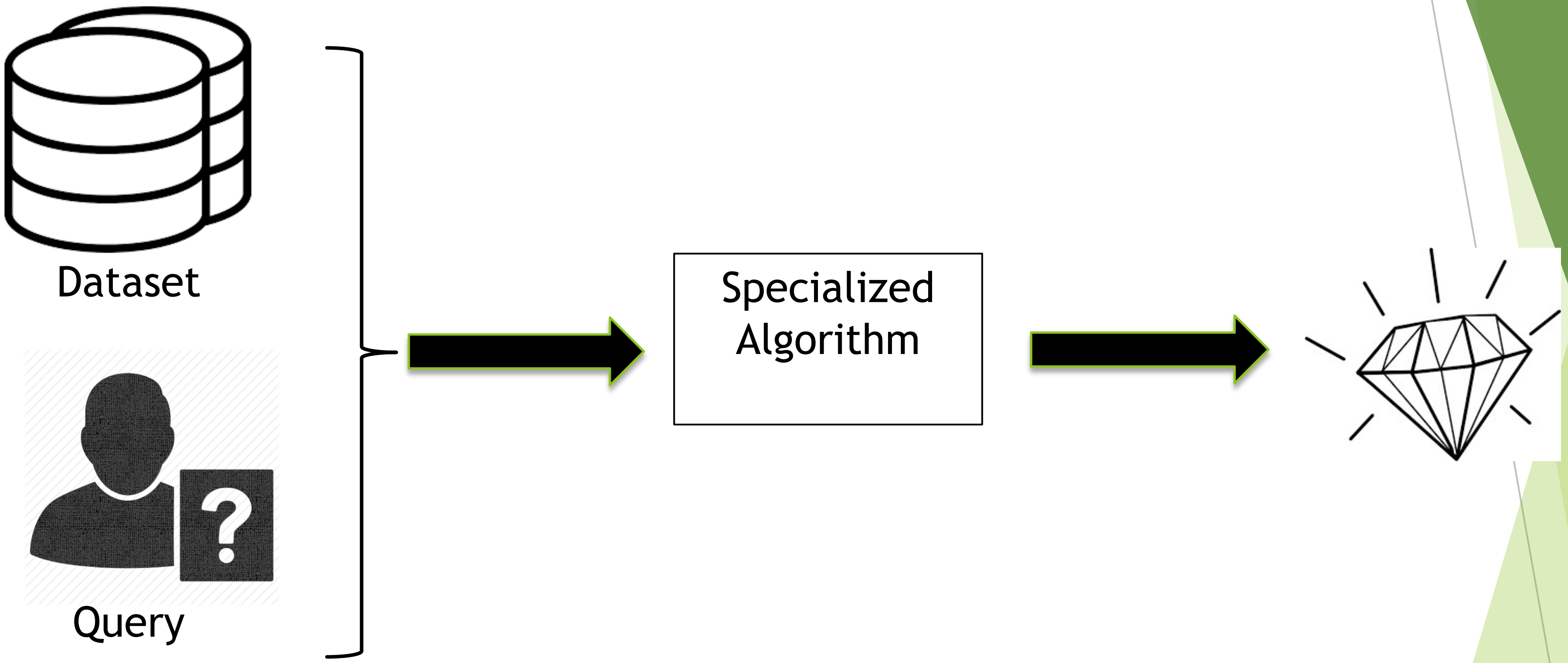
1 simula

2 LIRMM

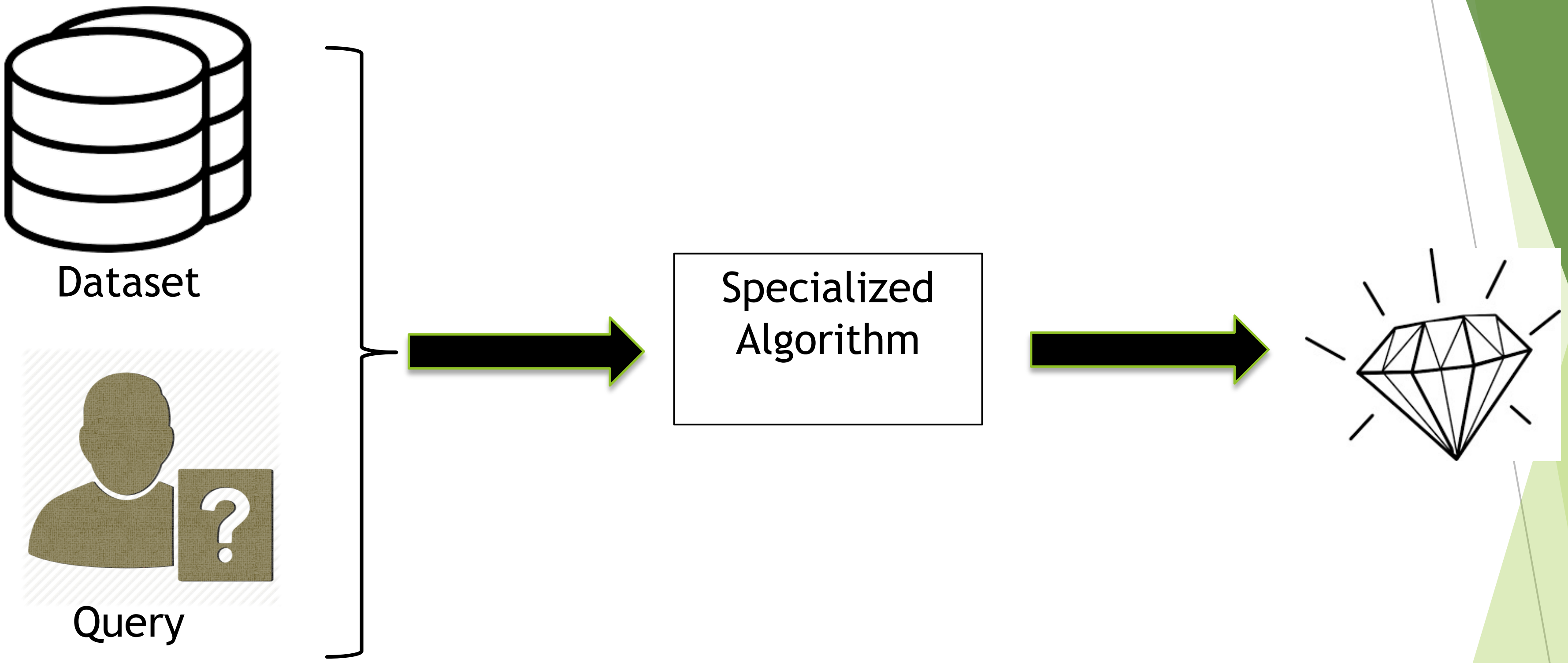
Introduction

- Aim: show the flexibility of CP to cope with additional dimension (multiple support)
- Can we do it? How? Is the propagation complete?
- What is the motivation? How can it be useful? Interesting queries?
- Accepted at ICTAI 2021

Motivation

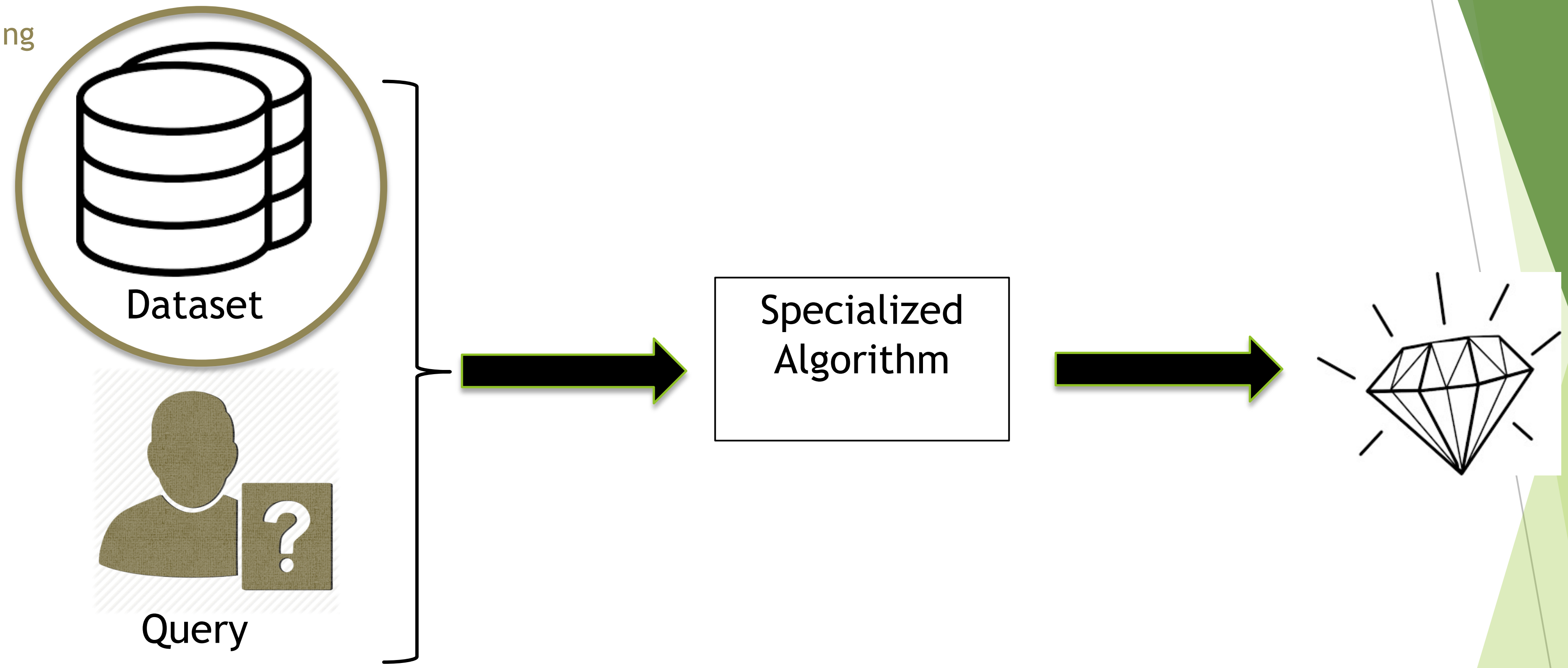


Motivation



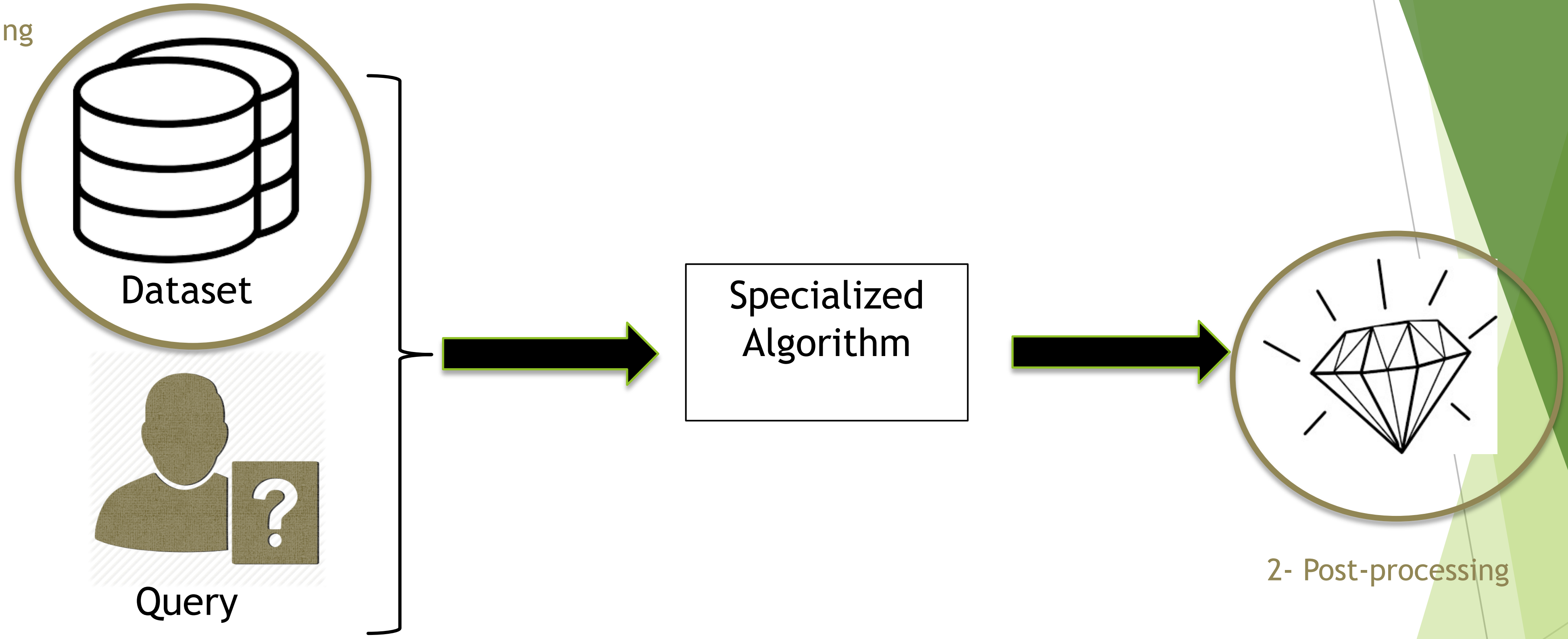
Motivation

1- Pre-processing

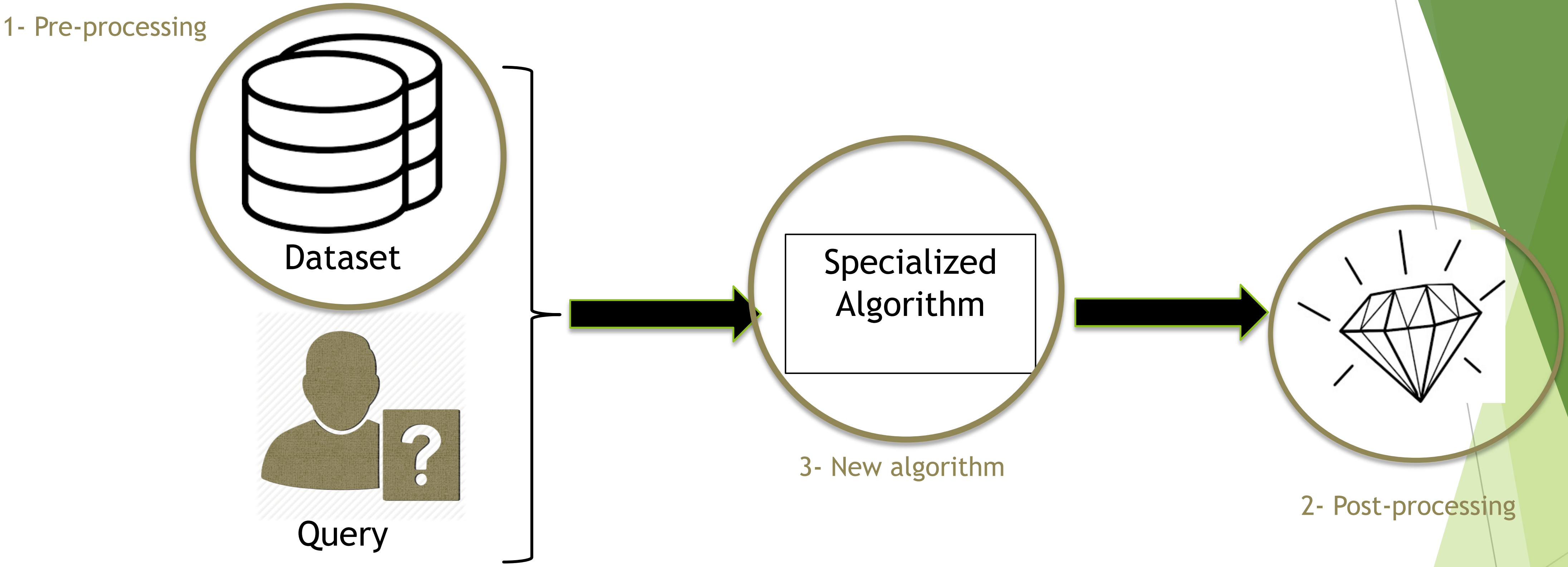


Motivation

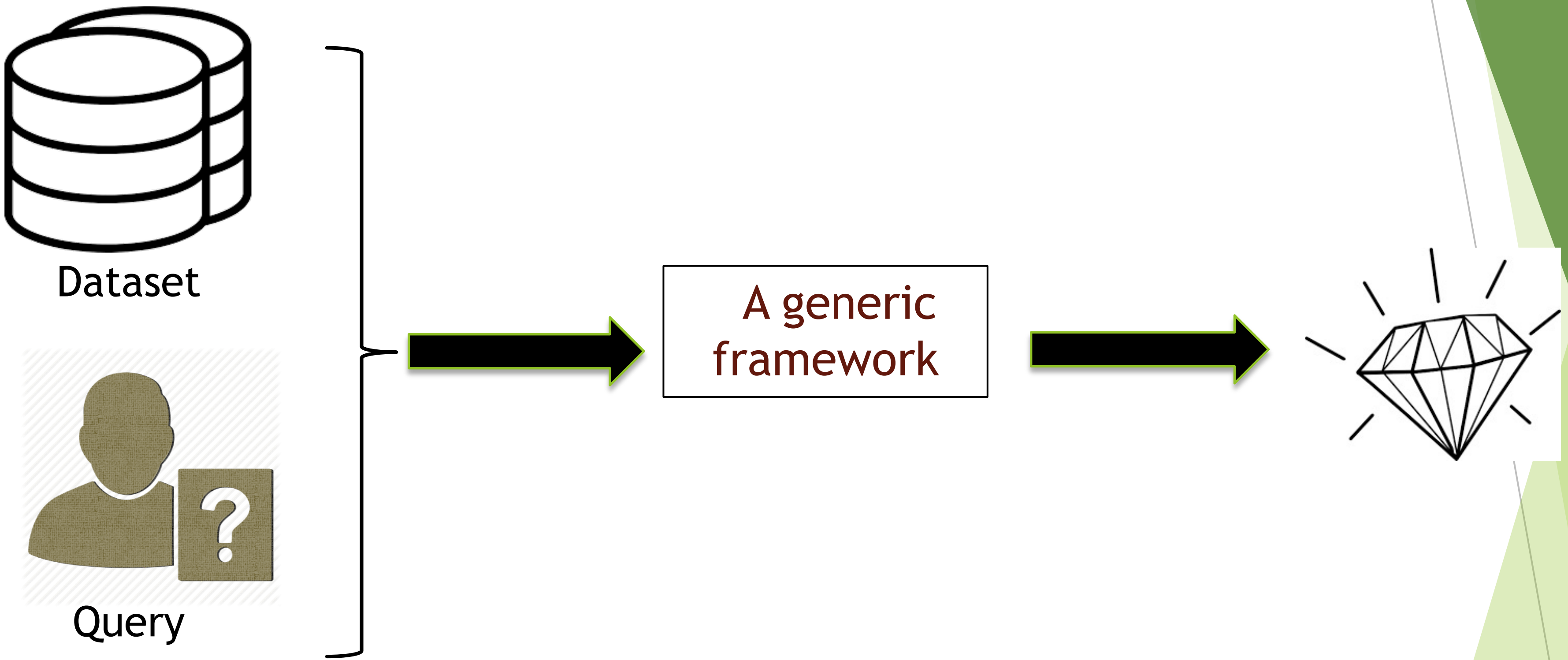
1- Pre-processing



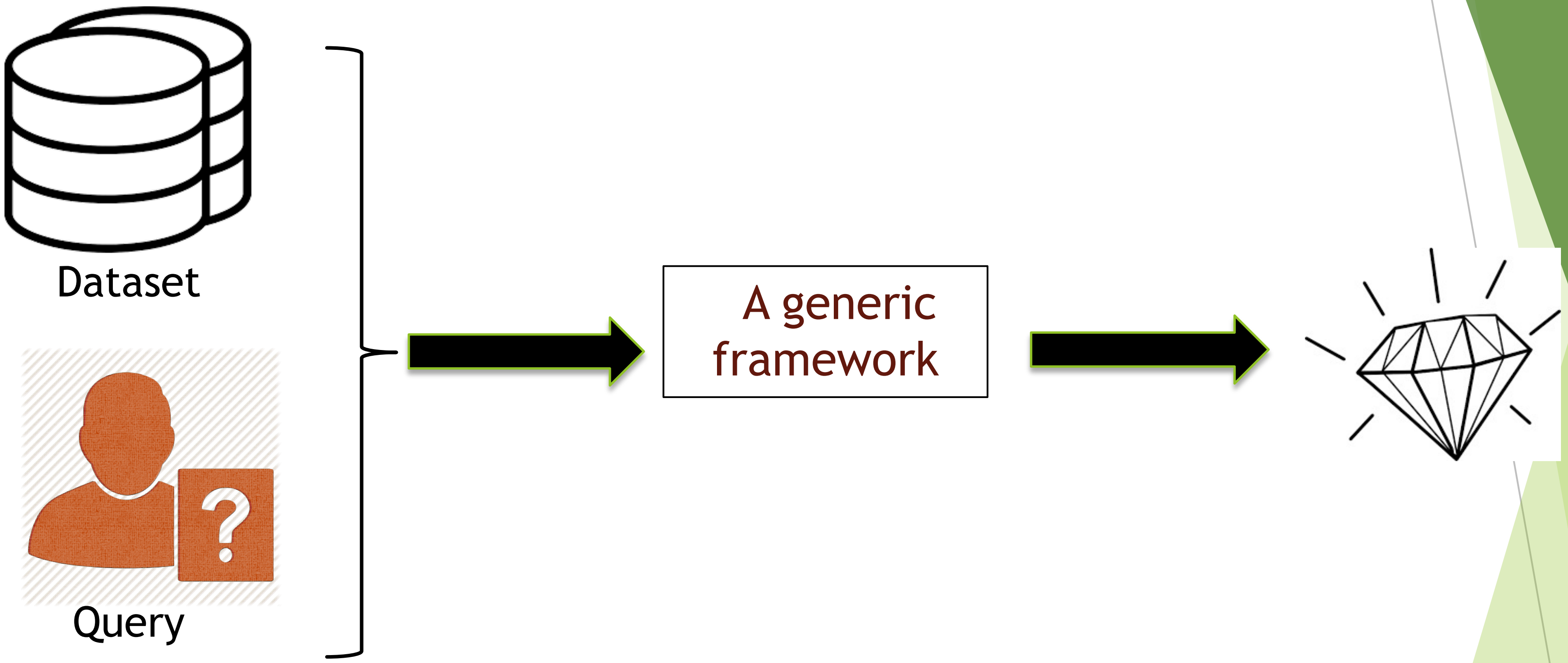
Motivation


1- Pre-processing
Dataset
Query
Specialized Algorithm
3- New algorithm
2- Post-processing

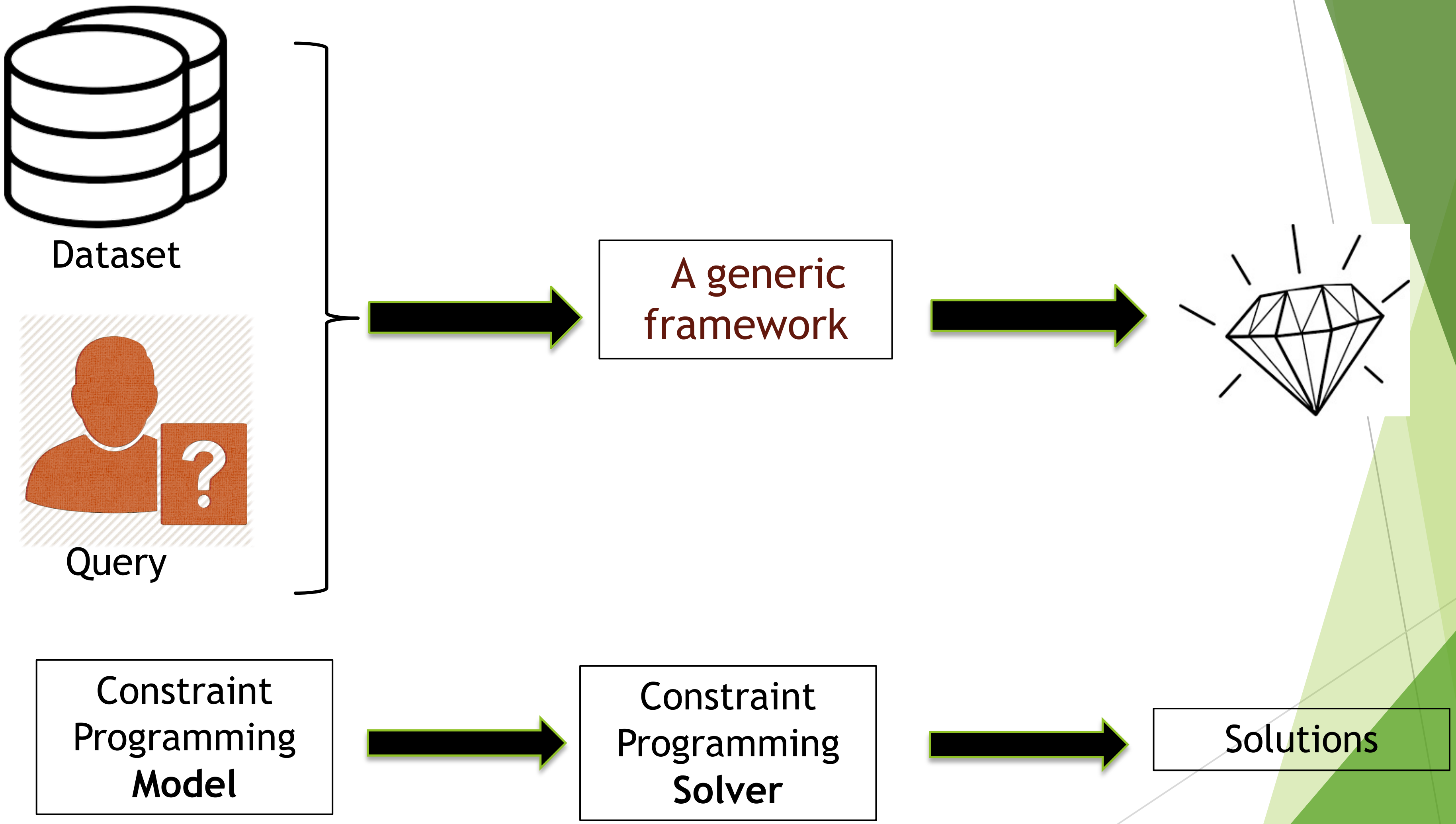
Motivation



Motivation



Motivation



Dataset
Query
A generic framework
Constraint Programming **Model**
Constraint Programming **Solver**
Solutions

Constraint Programming

Constraint Programming

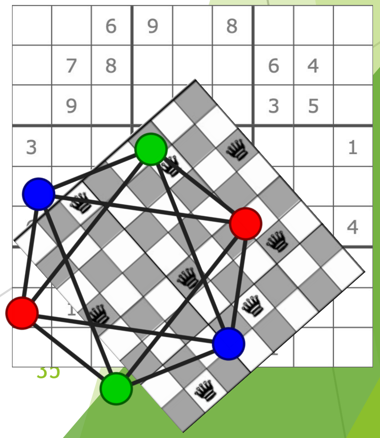
- In Constraint Programming (CP) the user declares:
  - A set of variables

$$X = \{x_1,\dots,x_n\}$$

  - A set of domains (set of possible values)

$$\mathbf{dom} = \{\mathbf{dom}(x_1),\dots,\ \mathbf{dom}(x_n)\}$$

  - A set of constraints **C** on variables where **c** is a relation between set of variables
- The constraint solver finds **solutions** (assignments on X satisfying all constraints)

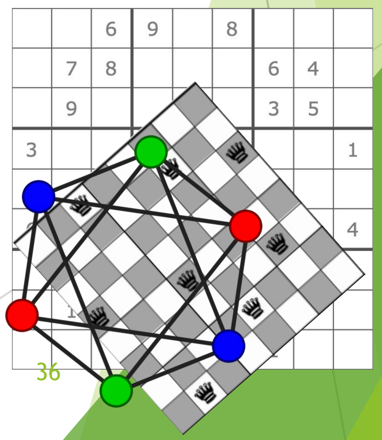


Constraint Programming

- A filtering algorithm (aka propagator)

$$\mathrm{dom}(X_1) = \{1,2,4\}$$
$$\mathrm{dom}(X_2) = \{2,3,5\}$$
$$\mathrm{dom}(X_3) = \{3,8,9\}$$

$$X_1 + X_2 = X_3$$

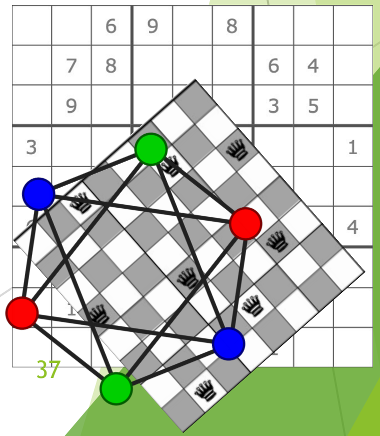


Constraint Programming

- A filtering algorithm (aka propagator)

dom(X_1) = {1,~~2~~,4}
dom(X_2) = {2,3,5}
dom(X_3) = {3,8,9}

$X_1 + X_2 = X_3$

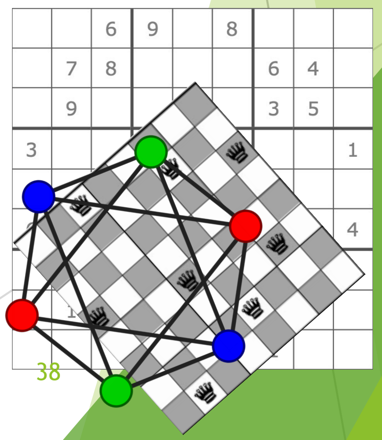


Constraint Programming

- A filtering algorithm (aka propagator)

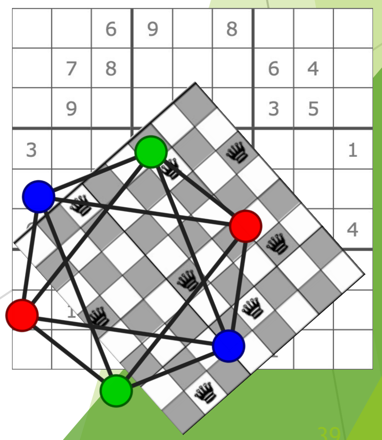
dom(X_1) = {1,~~2~~,4}
dom(X_2) = {2,~~3~~,5}
dom(X_3) = {3,~~8~~,9}

$X_1 + X_2 = X_3$



Global Constraints

- Constraints defined by a relation on any number of variables
- Example: **AllDifferent(x_1,…,x_n)** specifies that all its variables must take different values



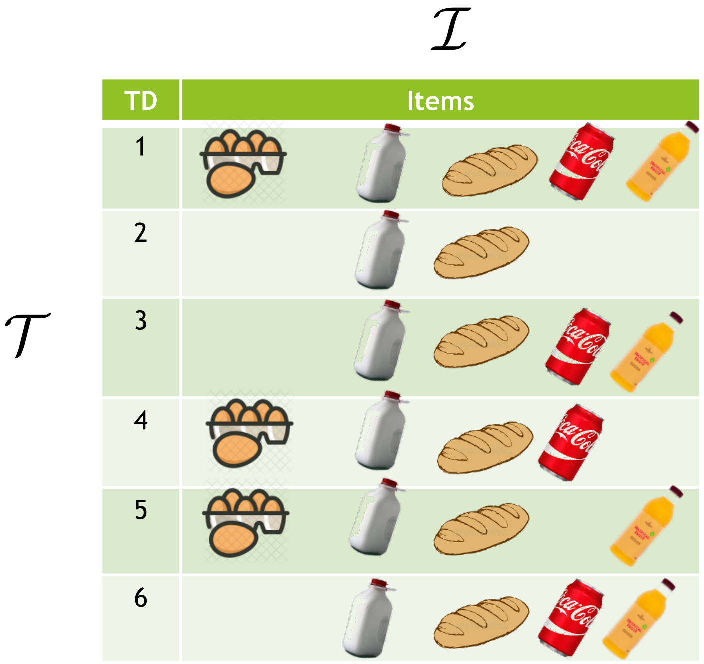
Itemset Mining

Itemset Mining

- Find useful patterns from transaction databases

Itemset Mining

- Find useful patterns from transaction databases



Frequent/Infrequent Itemsets

- Itemset = set of items
- Cover: cover(AB) = $\{t_1, t_4, t_5\}$
- Frequency: freq(AB) = |cover(AB)| = 3
- Given a frequency threshold s = 3:
  - AB is **frequent** (freq(AB) = 3 ≥ 3)
  - AD is **infrequent** (freq(AD) = 2 <3)

trans.	Items				
t_1	A	B	C	D	E
t_2		B	C		
t_3		B	C	D	E
t_4	A	B	C	D	
t_5	A	B	C		E
t_6		B	C	D	E

Frequent/Infrequent Itemsets

- Itemset = set of items
- Cover: cover(AB) = $\{t_1, t_4, t_5\}$
- Frequency: freq(AB) = |cover(AB)| = 3
- Given a frequency threshold s = 3:
  - AB is **frequent** (freq(AB) = 3 ≥ 3)
  - AD is **infrequent** (freq(AD) = 2 <3)

trans.	Items				
t_1	A	B	C	D	E
t_2		B	C		
t_3		B	C	D	E
t_4	A	B	C	D	
t_5	A	B	C		E
t_6		B	C	D	E

Frequent/Infrequent Itemsets

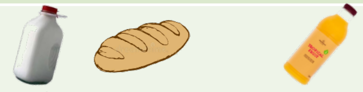
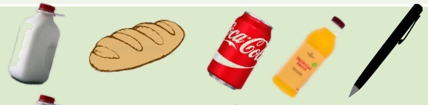
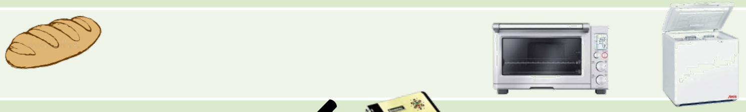
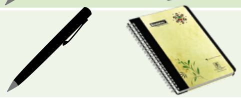
- Itemset = set of items
- Cover: cover(AB) = $\{t_1, t_4, t_5\}$
- Frequency: freq(AB) = |cover(AB)| = 3
- Given a frequency threshold s = 3:
  - AB is **frequent** (freq(AB) = 3 ≥ 3)
  - AD is **infrequent** (freq(AD) = 2 <3)

trans.	Items				
t_1	A	B	C	D	E
t_2		B	C		
t_3		B	C	D	E
t_4	A	B	C	D	
t_5	A	B	C		E
t_6		B	C	D	E

Single threshold problem (example)

$\mathcal{T}$

$\mathcal{I}$

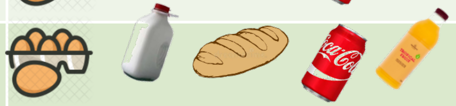
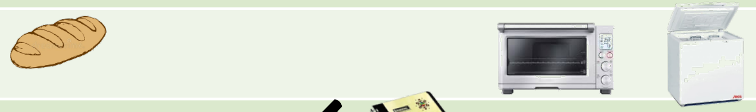
TD	Items
1	
2	
3	
4	
5	
6	
7	
8	

Single threshold problem (example)

$S = 4$

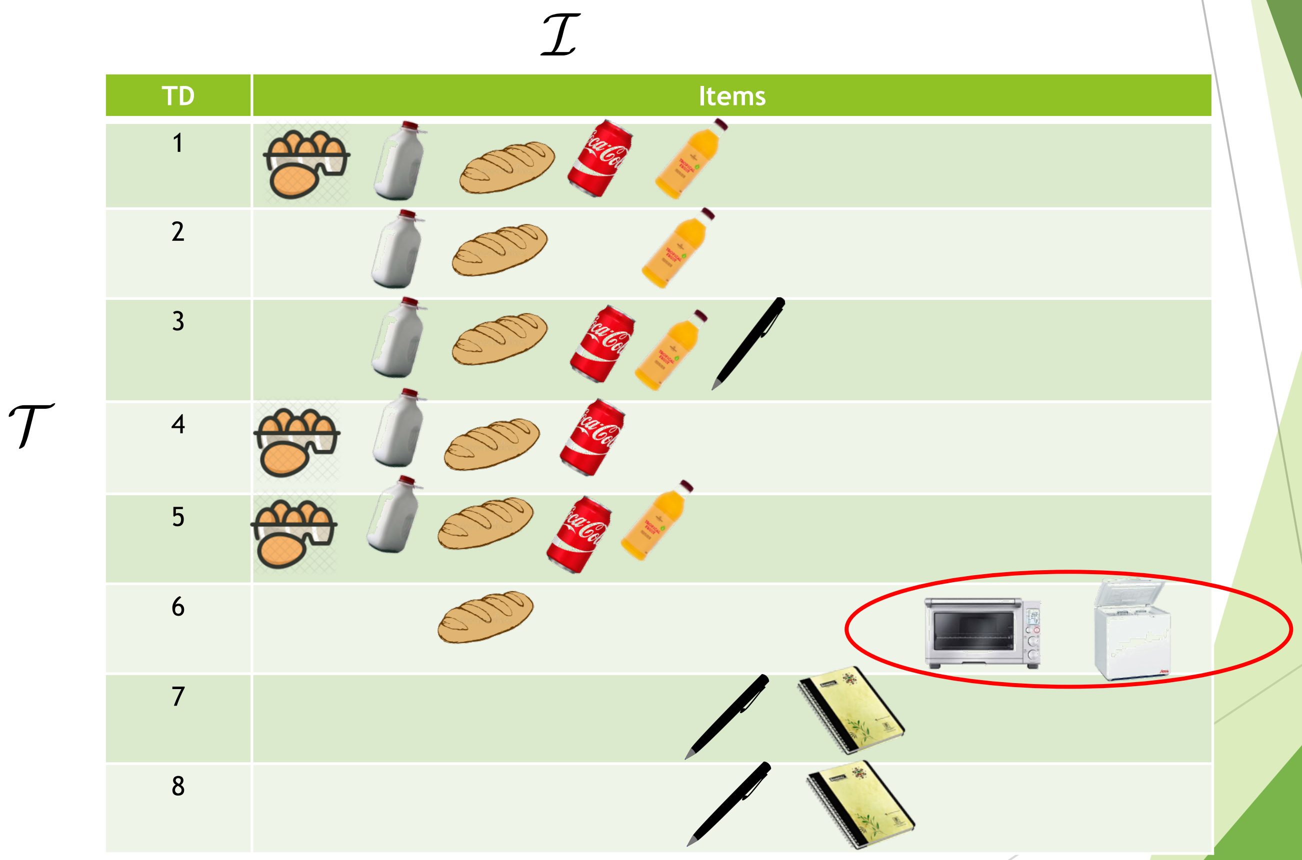
$\mathcal{I}$

$\mathcal{T}$

TD	Items
1	
2	
3	
4	
5	
6	
7	
8	

Single threshold problem (example)

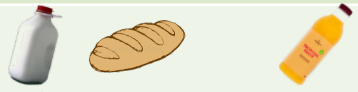
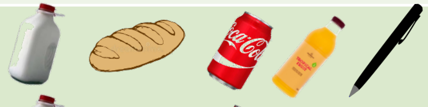
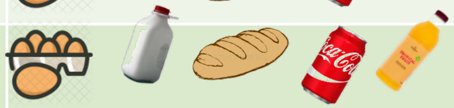
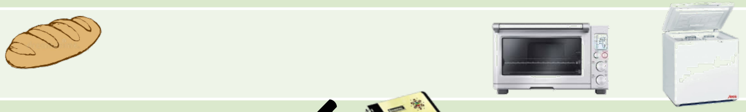
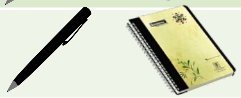
S = 4



Single threshold problem (example)

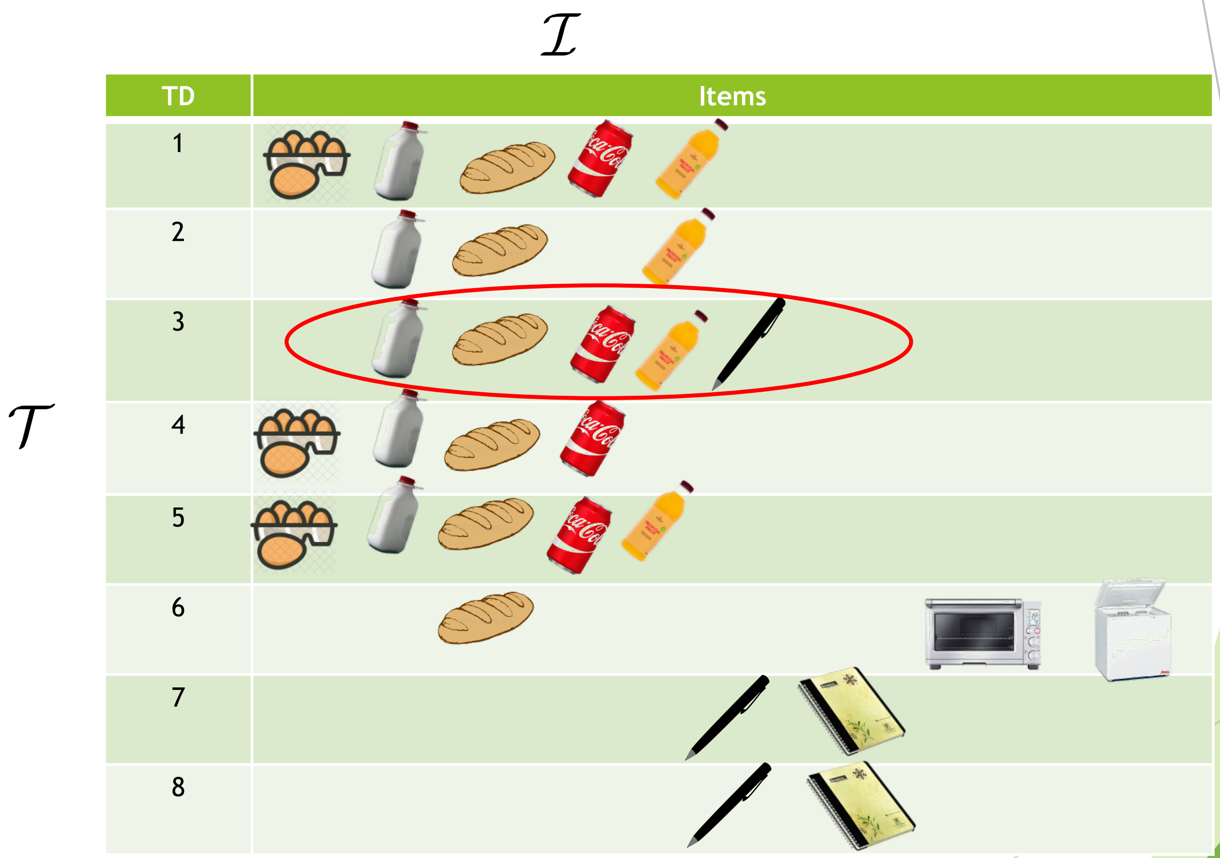
$\mathcal{I}$

$\mathcal{T}$

TD	Items
1	
2	
3	
4	
5	
6	
7	
8	

S = 1

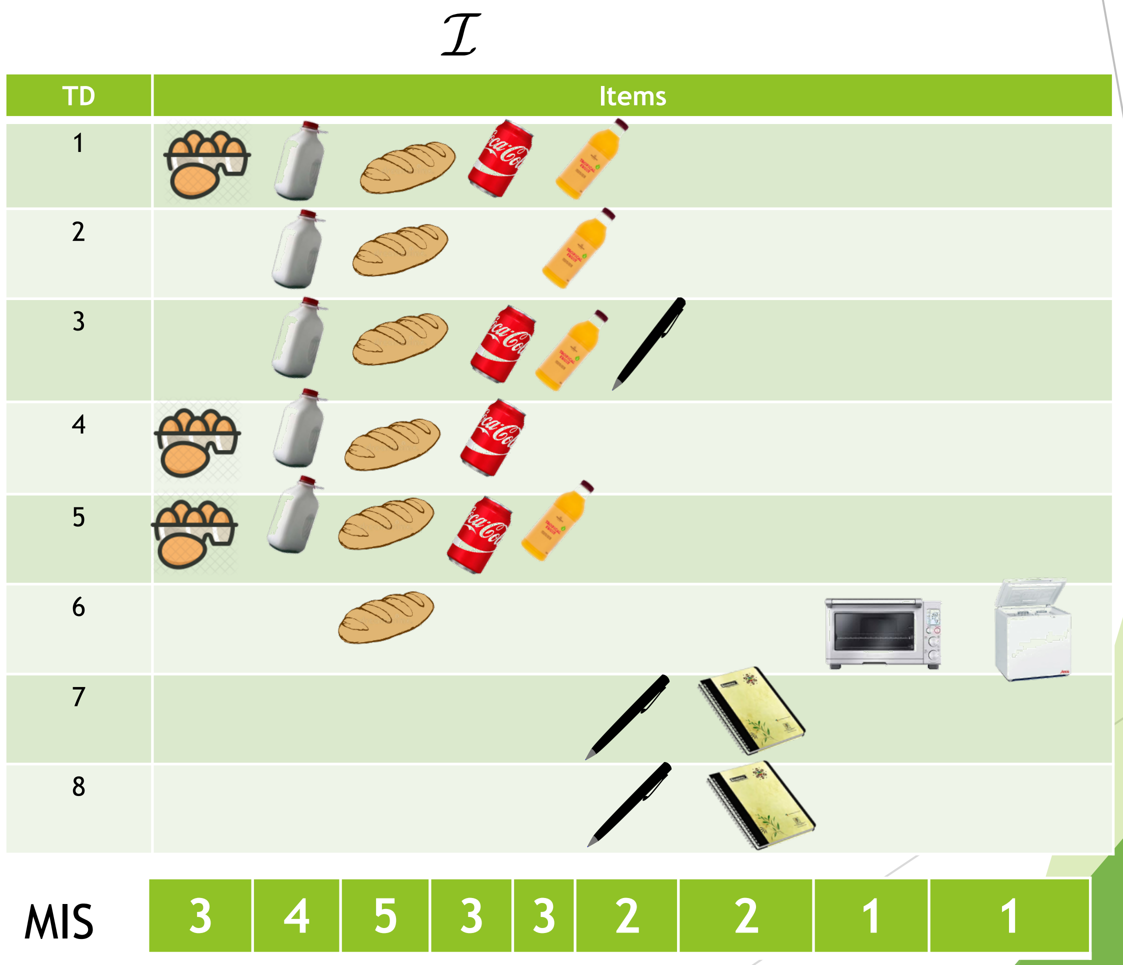
Single threshold problem (example)



S = 1

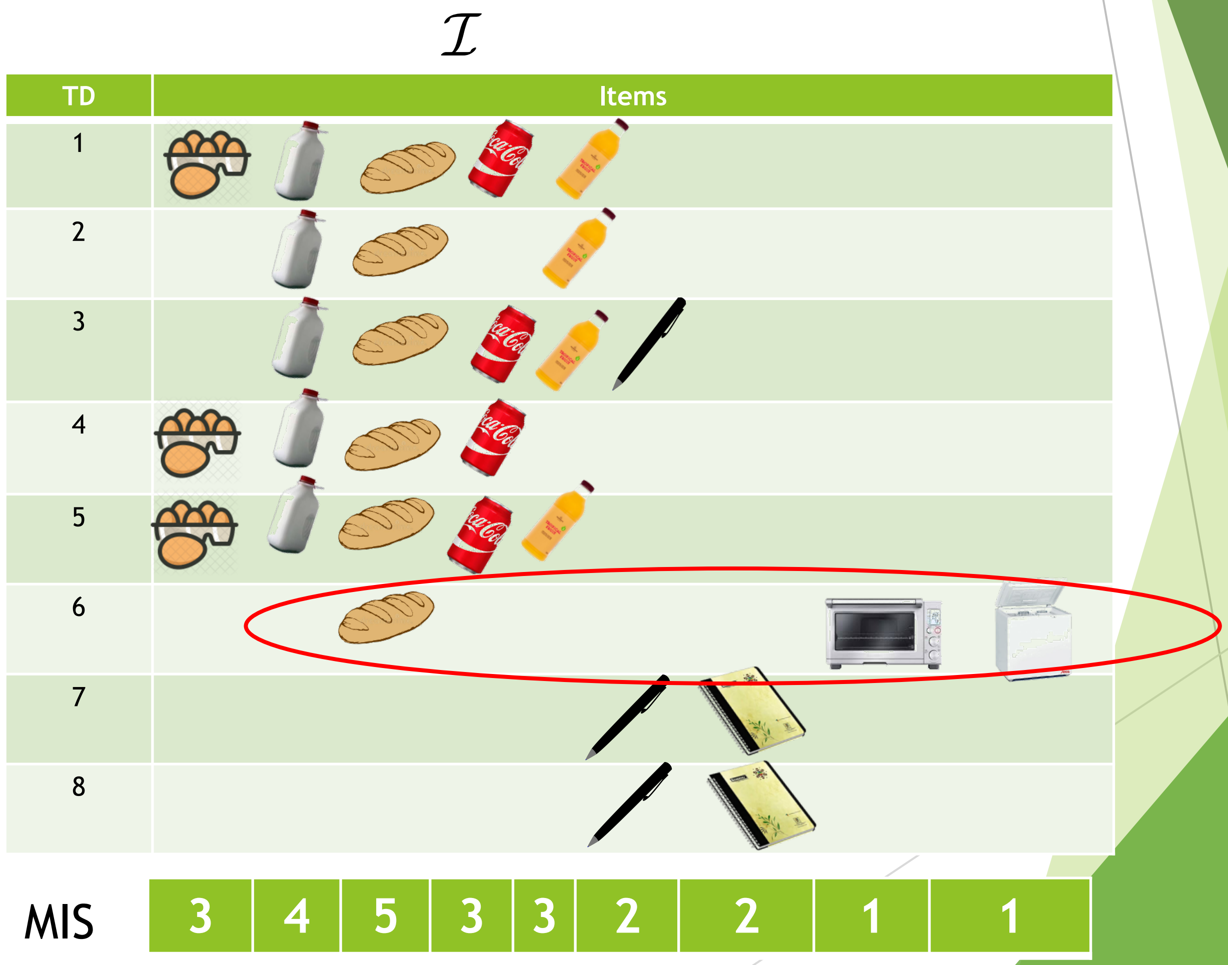
Single threshold problem (example)

$\mathcal{T}$

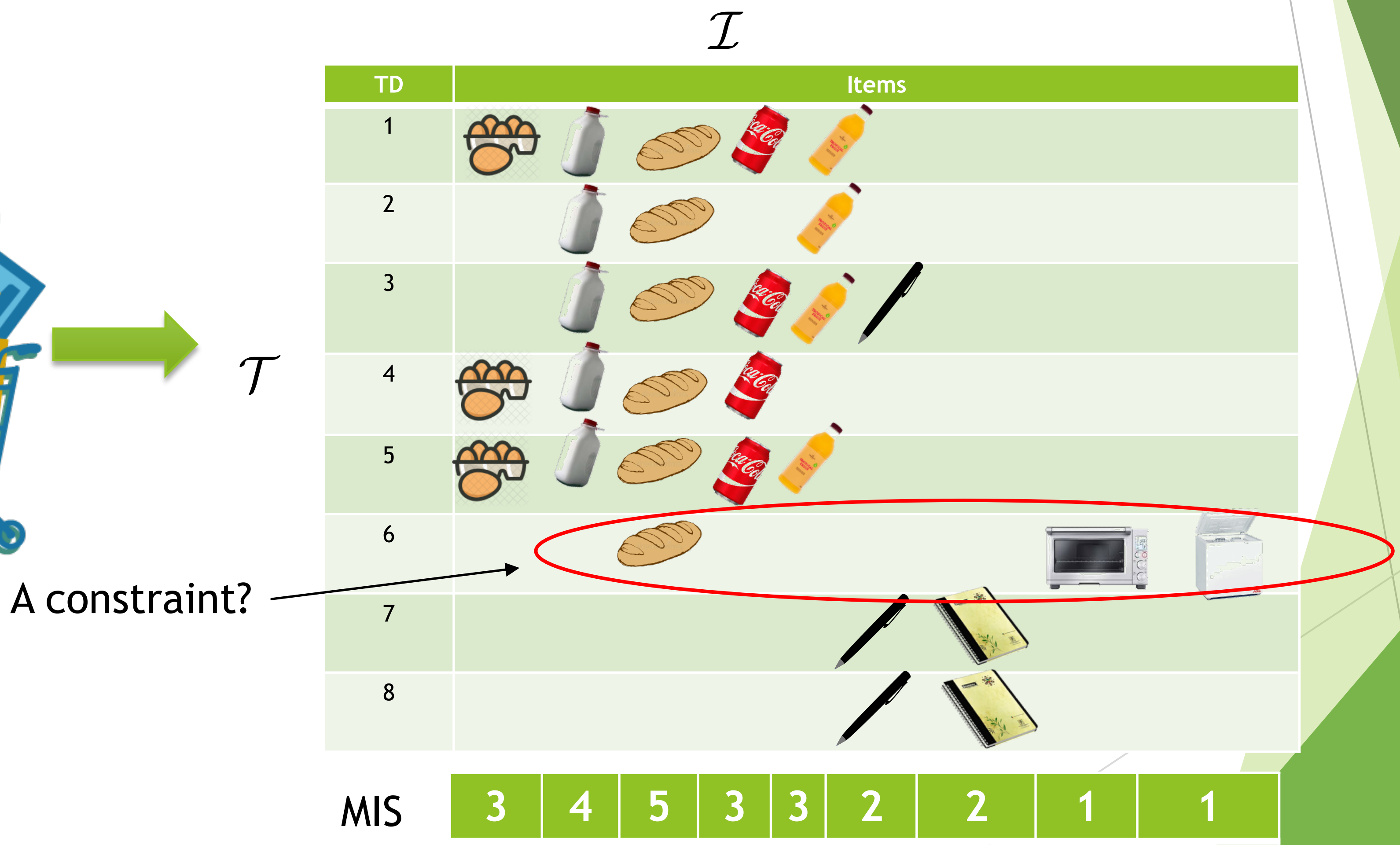


Single threshold problem (example)

$\mathcal{T}$



Single threshold problem (example)



Basic CP model for mining frequent itemsets (Luc De Raedt et.al, 2008)

- Variables:
  - A binary variable for every item **i**: the presence of the item **i** in the searched itemset (P)
  - A binary variable for every transaction **t**: the presence of the searched itemset (P) in the transaction **t**
- Constraints (reified):
  - Cover constraint
  - Threshold constraint (freq(P) $\geq$ s)

With multiple minimum supports (MIS)

- Extend the model:
  - Replace “freq(P) ≥ s” by “freq(P) ≥ min(MIS_k|k in P)”
  - Does not scale!
- Define a global constraint “FreqRare”:
  - Only item variables (no need for transaction variables)
  - Dedicated propagator

FreqRare

- Holds if the searched itemset (P={i|x_i=1}) is frequent w.r.t the list MIS
- Propagator → remove 1 from x_i if including i results a frequency less than the minimum of remaining MIS values
- Time complexity: O(|items|*|transactions|)
- Result → Backtrack-free using minimum MIS as variable ordering heuristic

User queries

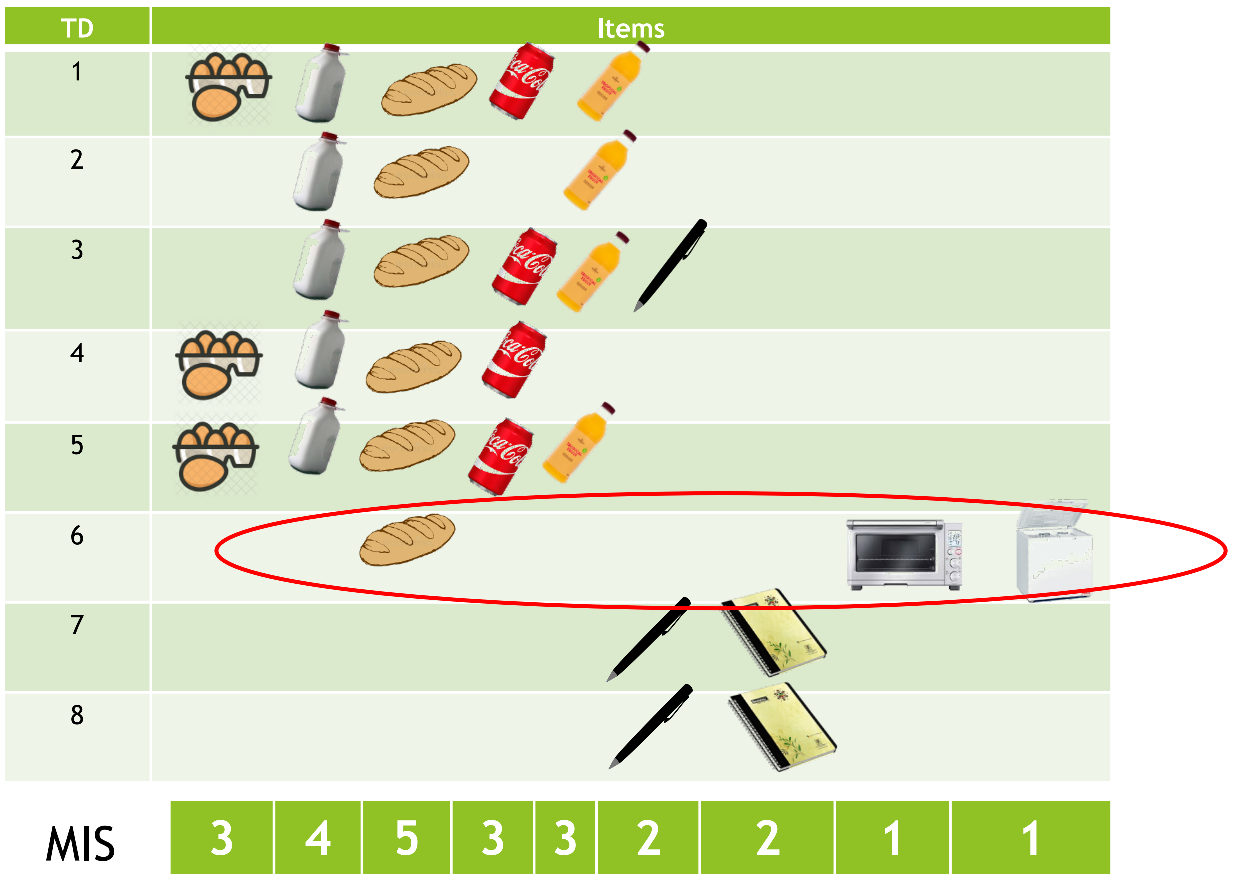
- In CP → simply extend the model
- Specialized methods → a post processing step (checker)

User queries

- Return itemsets including items of the same type (distance between MISs is bounded above):
  - $|MIS\_i - MIS\_j| \leq ub$
- Size of the itemset is bounded below:
  - $|P| \geq c$
- K-pattern mining [Guns et al., 2011] (K patterns with constraints between them):
  - K vectors of Boolean variables
  - K distinct itemsets satisfying both constraints

User queries

$\mathcal{I}$



User queries

- Return itemsets including items of the same type (distance between MISs is bounded above):
  - $|MIS\_i - MIS\_j| \leq ub$
- Size of the itemset is bounded below:
  - $|P| \geq c$
- K-pattern mining [Guns et al., 2011] (K patterns with constraints between them):
  - K vectors of Boolean variables
  - K distinct itemsets satisfying both constraints

Experiments

- We selected several real-sized datasets from the FIMI repository
- Our approach (**CP4MIS**) compared with: 1) CPFGrowth++ (SPMF implementation) 2) Basic CP Model (Rmodel)
- For CP we have used **Oscar solver** within Scala
- MIS_i = max(Beta*freq(i), Min) as in [Bing Liu et.al, 1999]
- Machine = Intel core **i7**, 2.8Ghz with a RAM of **16GB**
- Time limit = one hour

Results (Mining frequent itemsets)

Q0:	CFPG (a)	Rmodel (b)		CP4MIS (c)		#sol
	Time	Time	Memory	Time	Memory	
Zoo	**0.81**	12.00	3,760	1.34	**20**	1.3M
Vote	**1.56**	196.17	2,164	2.23	**8**	2.1M
Anneal	**30.91**	134.74	3,095	64.82	**49**	71.7M
Chess	**11.64**	305.03	3,153	28.20	**67**	22.6M
Mushroom	**45.53**	TO	–	106.00	**48**	105.2M
Connect	**48.45**	TO	–	854.59	**218**	91.7M
T40	409.55	–	OOM	**91.70**	**2,304**	15.8M
Pumsb	**38.60**	–	OOM	115.67	**916**	13.5M

Results (Mining frequent itemsets)

	CFPG	Rmodel		CP4MIS		
Q0:	(a)	(b)		(c)		#sol
	Time	Time	Memory	Time	Memory	
Zoo	**0.81**	12.00	3,760	1.34	**20**	1.3M
Vote	**1.56**	196.17	2,164	2.23	**8**	2.1M
Anneal	**30.91**	134.74	3,095	64.82	**49**	71.7M
Chess	**11.64**	305.03	3,153	28.20	**67**	22.6M
Mushroom	**45.53**	TO	–	106.00	**48**	105.2M
Connect	**48.45**	TO	–	854.59	**218**	91.7M
T40	409.55	–	OOM	**91.70**	**2,304**	15.8M
Pumsb	**38.60**	–	OOM	115.67	**916**	13.5M

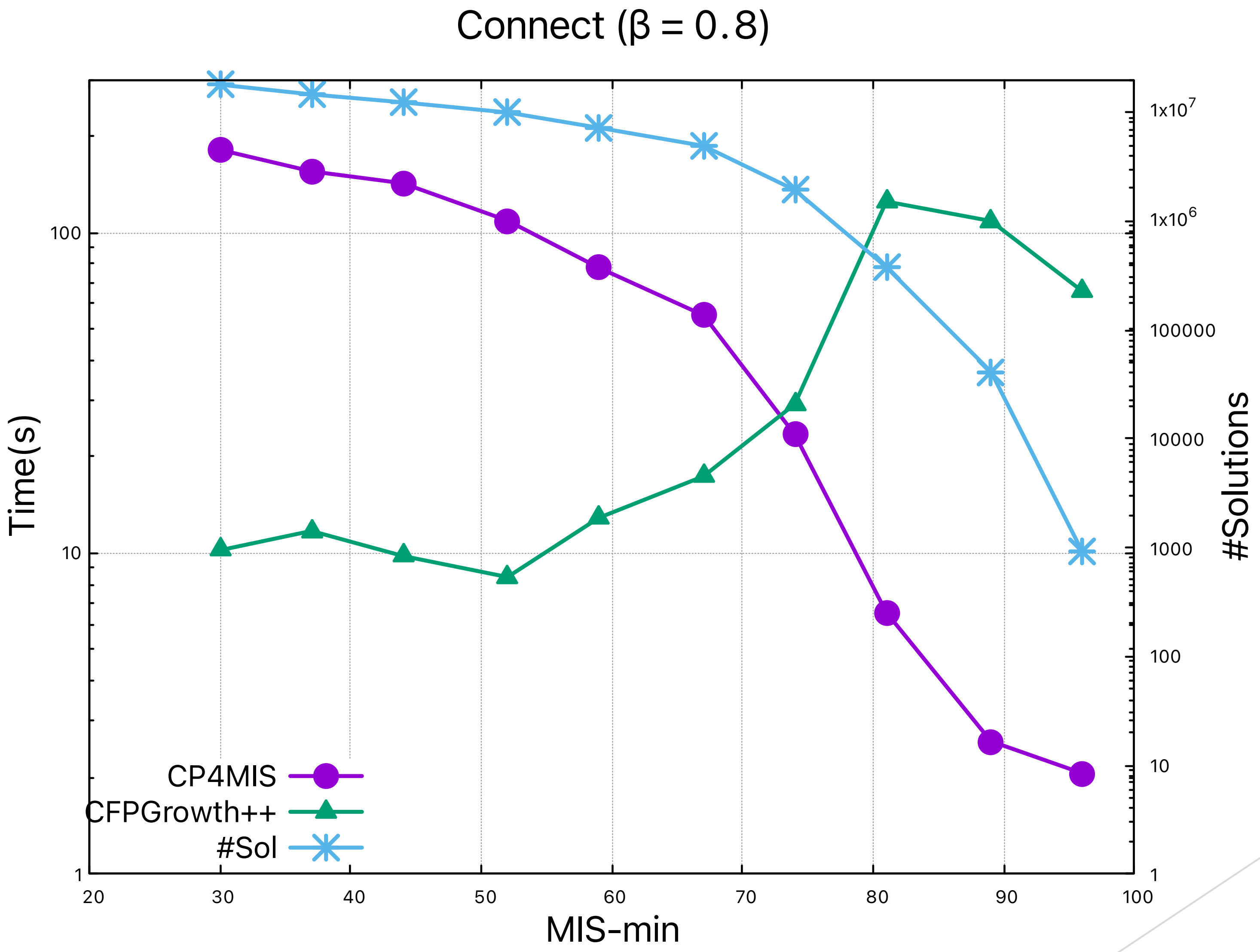
Results (Mining frequent itemsets)

Q0:	CFPG (a)	Rmodel (b)		CP4MIS (c)		#sol
	Time	Time	Memory	Time	Memory	
Zoo	**0.81**	12.00	3,760	1.34	**20**	1.3M
Vote	**1.56**	196.17	2,164	2.23	**8**	2.1M
Anneal	**30.91**	134.74	3,095	64.82	**49**	71.7M
Chess	**11.64**	305.03	3,153	28.20	**67**	22.6M
Mushroom	**45.53**	TO	–	106.00	**48**	105.2M
Connect	**48.45**	TO	–	854.59	**218**	91.7M
T40	409.55	–	OOM	**91.70**	**2,304**	15.8M
Pumsb	**38.60**	–	OOM	115.67	**916**	13.5M

Results (Mining frequent itemsets)

	CFPG	Rmodel		CP4MIS		
Q0:	(a)	(b)		(c)		#sol
	Time	Time	Memory	Time	Memory	
Zoo	**0.81**	12.00	3,760	1.34	**20**	1.3M
Vote	**1.56**	196.17	2,164	2.23	**8**	2.1M
Anneal	**30.91**	134.74	3,095	64.82	**49**	71.7M
Chess	**11.64**	305.03	3,153	28.20	**67**	22.6M
Mushroom	**45.53**	TO	–	106.00	**48**	105.2M
Connect	**48.45**	TO	–	854.59	**218**	91.7M
T40	409.55	–	OOM	**91.70**	**2,304**	15.8M
Pumsb	**38.60**	–	OOM	115.67	**916**	13.5M

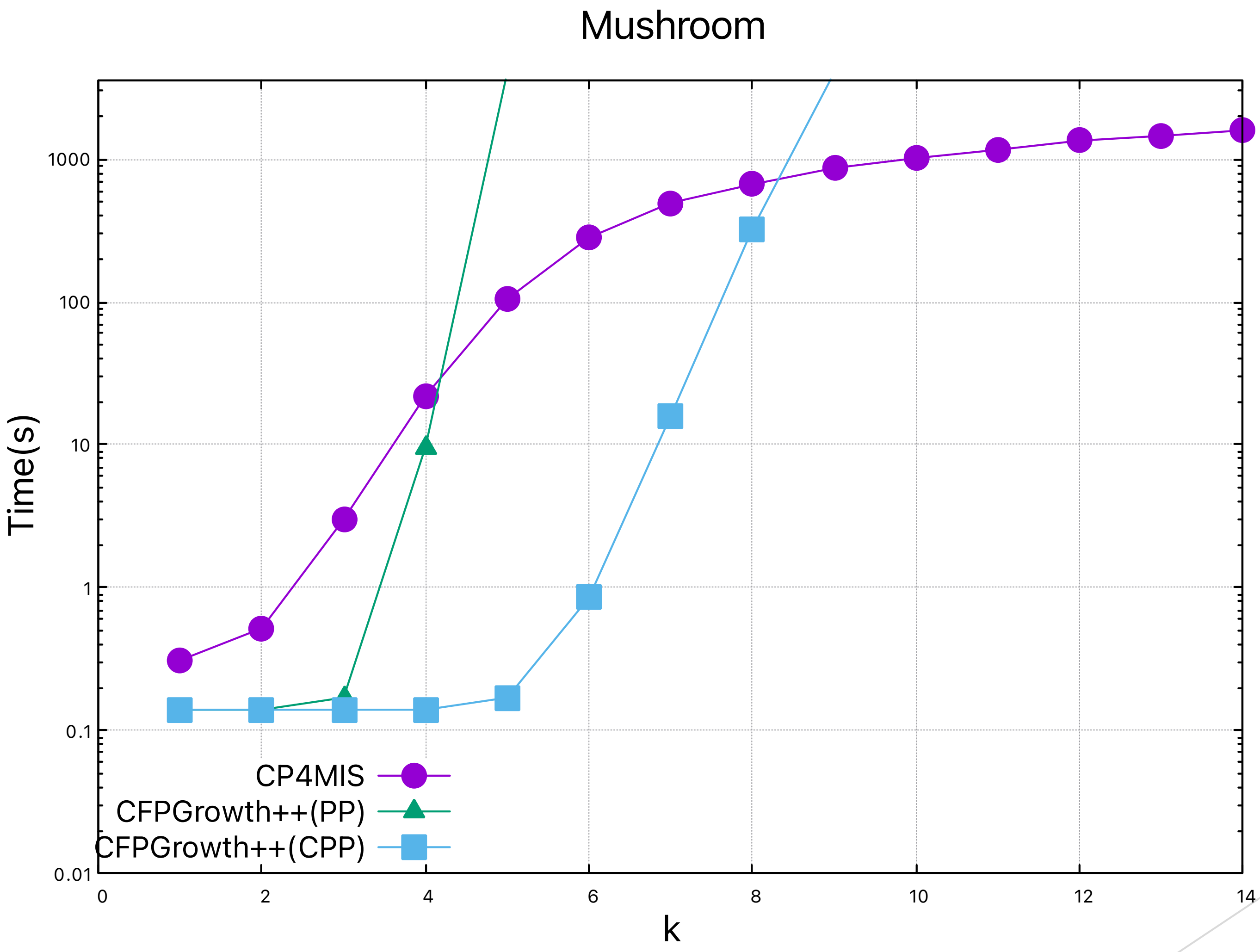
Results (CFPG vs CP4MIS)



Results (Constrained itemsets)

			CFPG +Checker	CP4MIS	
Q2:	ub	c	(d)	(c)	#sol
Zoo	2	10	0.62	**0.10**	14
Vote	1	10	1.14	**0.13**	12
Anneal	30	8	12.78	**0.18**	7
Chess	80	8	6.49	**0.23**	30
Mushroom	50	8	19.19	**0.36**	27
Connect	1000	10	20.88	**2.03**	2
T40	100	6	389.80	**54.31**	14
Pumsb	1000	8	27.91	**2.86**	17

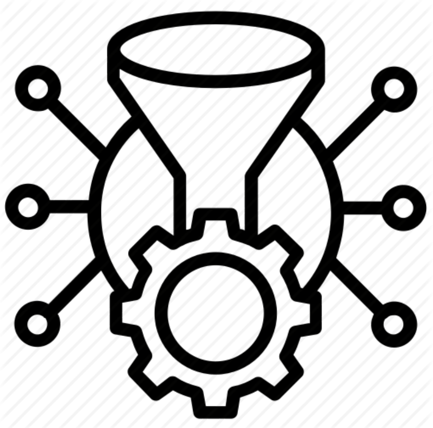
Results (K-pattern mining)



Conclusion

- We have introduced a CP-based approach for mining frequent itemsets with multiple minimum supports
- We have provided a **propagator** and showed that, using minMIS heuristic, the propagation is **backtrack-free** (0 fails)
- Our CP approach have shown the flexibility and the performance in taking in consideration additional user constraints
- Future: use the expressiveness of CP to solve problems that involve more complex constraints on MISs

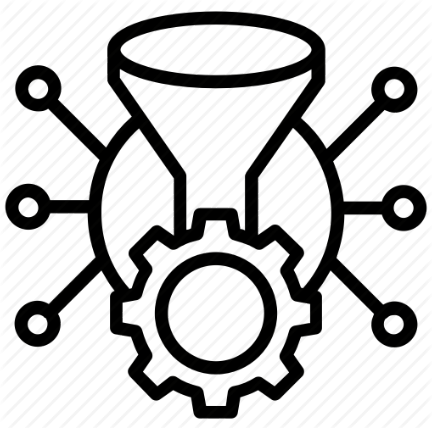
t1			T			S	I	O		U	
t2	A	N	T	H	K	S			E		
t3	A				K	S	I				Q



bachir@simula.no

t1			T			S	I	O		U	
t2	A	N	T	H	K	S			E		
t3	A				K	S	I				Q

THANKS



bachir@simula.no