EpisodeSupport: a Global Constraint for Mining Frequent Patterns in a Long Sequence of Events

Q. Cappart[1], **J. AOGA**[1], P. Schaus[1]

[1]UCLouvain — Belgium

CPAIOR 2018, Delft, The Netherlands, 26–29/06/2018

Context

- This talk is about finding **frequent episode patterns** in a **long (time-stamped) sequence**
  - Very efficient dedicated algorithms exists (Minepi, Winepi, Emma,…)
    - They are not **flexible** and suffer for **memory problem**
- Motivation for CP:
  - finding **frequent (constrained) sub-sequences** is a related problem to **frequent (constrained) episode patterns** in a long (time-stamped) sequence
    - constraint example: satisfying a regular constraint
  - CP-based method is the state-of-the-art for finding **frequent (constrained) sub-sequences** in a sequence database *[Aoga et al., ECMLPKDD'16; CPAIOR'17]*

SPM Problem

Sub sequence **Sequence**

Client1	Milk	**Coffee**	Sugar	Coffee	Sugar
Client2	Coffee	Milk	Coffee	Sugar	
Client3	Milk	Coffee			
Client4	Coffee	Sugar	Egg		

Sequence Database (SDB)

- Sequence : < Milk Coffee Sugar Coffee Sugar>
- Subsequence : <Coffee Sugar>
- Freq (<Coffee Sugar>) = 3

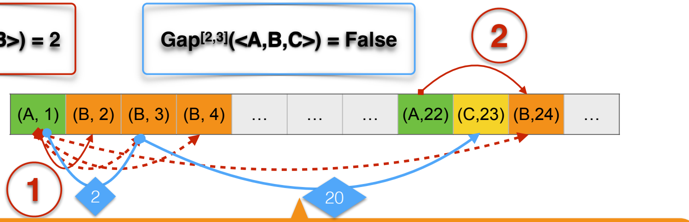
Problem : Find all subsequences with support ≥ Given Threshold

(Constraint-Based) Episode Mining problem

(Symbol, position)

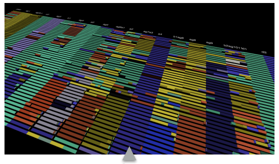
Freq(<A,B>) = 2

Gap$^{[2,3]}$(<A,B,C>) = False



Problem : Find All Episodes wrt. user-defined constraints (e.g. support ≥ Given Threshold)

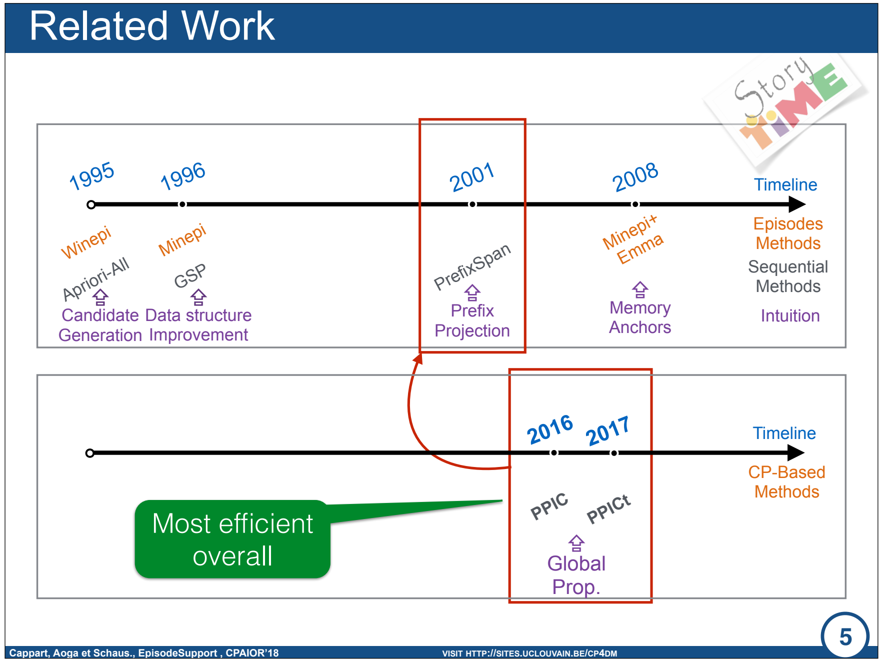
☑ Applications



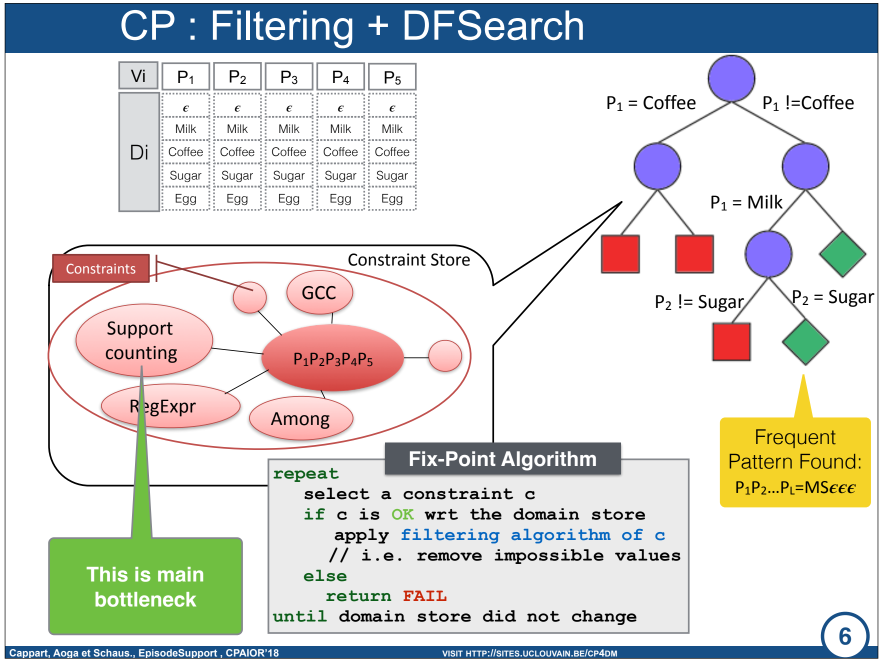
DNA Sequence

Smartphone lifelogging

Stock market



Related Work

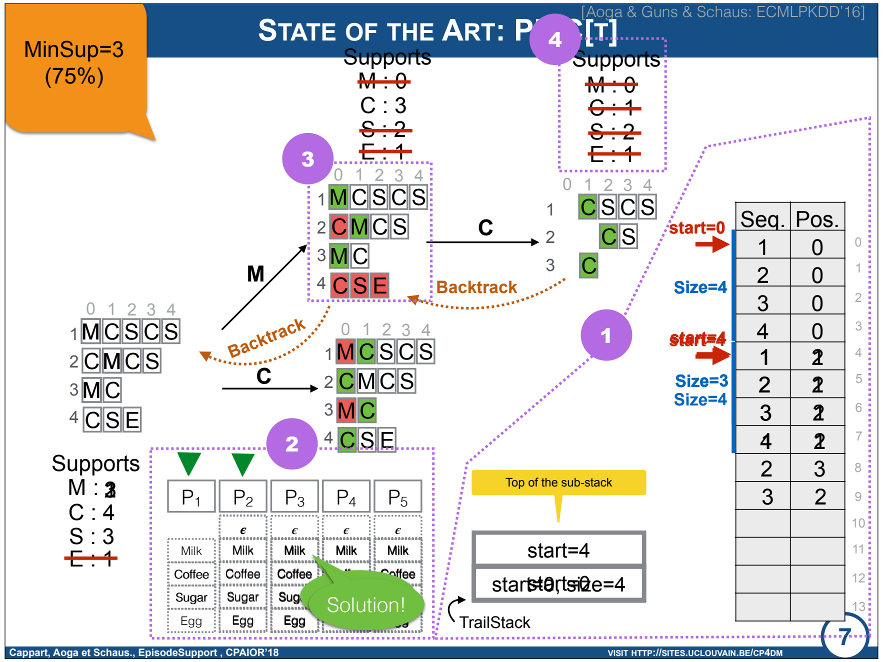


CP : Filtering + DFSearch

Vi	P_1	P_2	P_3	P_4	P_5
Di	ϵ	ϵ	ϵ	ϵ	ϵ
	Milk	Milk	Milk	Milk	Milk
	Coffee	Coffee	Coffee	Coffee	Coffee
	Sugar	Sugar	Sugar	Sugar	Sugar
	Egg	Egg	Egg	Egg	Egg

Fix-Point Algorithm

```
repeat
   select a constraint c
   if c is OK wrt the domain store
      apply filtering algorithm of c
      // i.e. remove impossible values
   else
     return FAIL
until domain store did not change
```

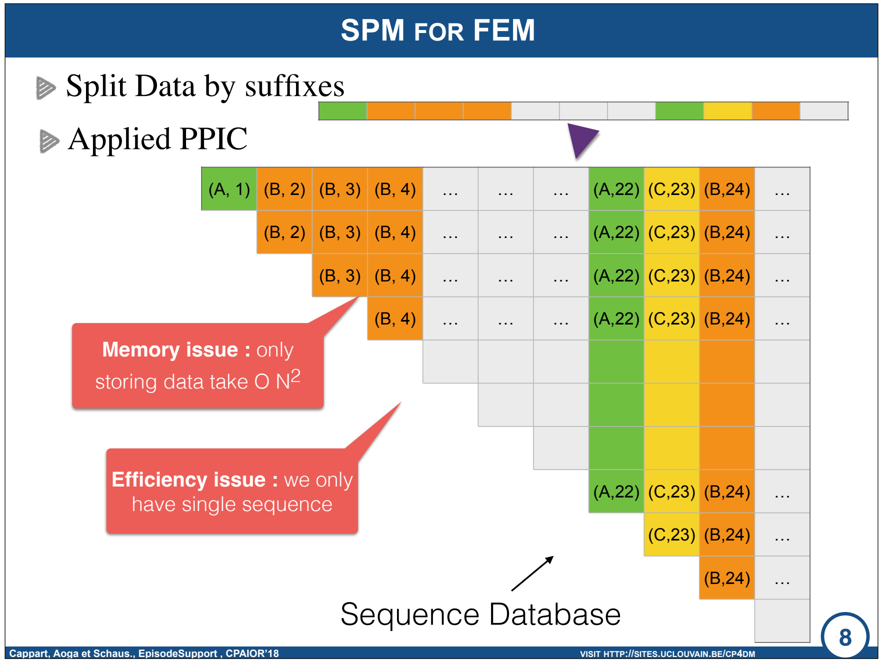


State of the Art: PPIC[T]

[Aoga & Guns & Schaus: ECMLPKDD'16]

MinSup=3 (75%)

Seq.	Pos.
1	0
2	0
3	0
4	0
1	2
2	2
3	2
4	2
2	3
3	2



SPM for FEM

- Split Data by suffixes
- Applied PPIC

(A, 1)	(B, 2)	(B, 3)	(B, 4)	…	…	…	(A,22)	(C,23)	(B,24)	…
	(B, 2)	(B, 3)	(B, 4)	…	…	…	(A,22)	(C,23)	(B,24)	…
		(B, 3)	(B, 4)	…	…	…	(A,22)	(C,23)	(B,24)	…
			(B, 4)	…	…	…	(A,22)	(C,23)	(B,24)	…
							(A,22)	(C,23)	(B,24)	…
								(C,23)	(B,24)	…
									(B,24)	…

Memory issue : only storing data take O N^2

Efficiency issue : we only have single sequence

Sequence Database

EpisodeSupport : Contribution

CONTRIBUTION

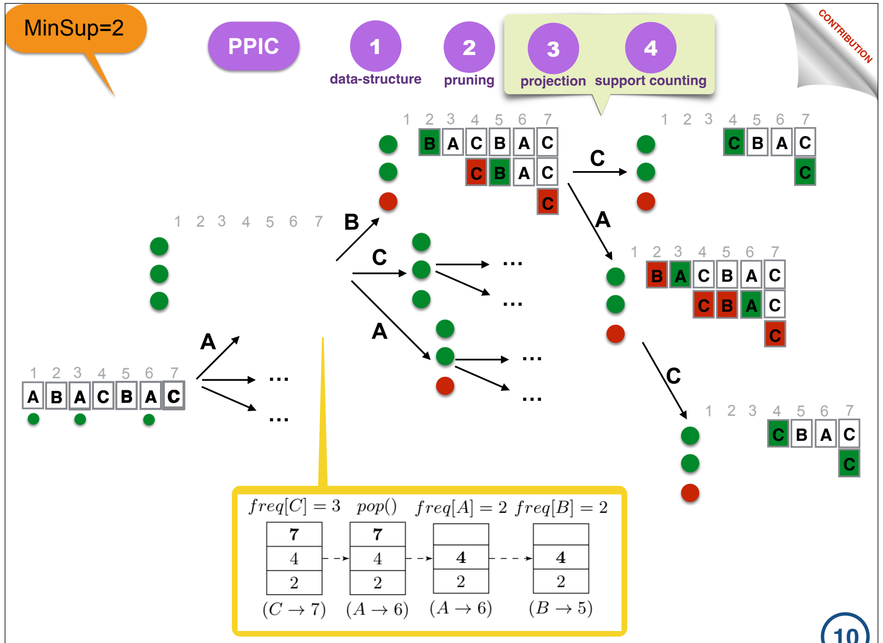
Goal: Design new Approach for finding Episodes capturing the most common constraints (including syntax and time-related constraints)

☑ *Adapt* trailed-based data structure to efficiently overcome memory issue

☑ Take into account that we have a single sequence with algorithmic improvements

☑ Tackle time series data and time-related constraints

☑ Show real application handling many other contraints: Regular/ Grammar, Gcc, Among, …



$freq[C] = 3$ $pop()$ $freq[A] = 2$ $freq[B] = 2$

7	**7**		
4	4	**4**	**4**
2	2	2	2
$(C \to 7)$	$(A \to 6)$	$(A \to 6)$	$(B \to 5)$

Experiments

OSCAR www.oscarlib.org Scala

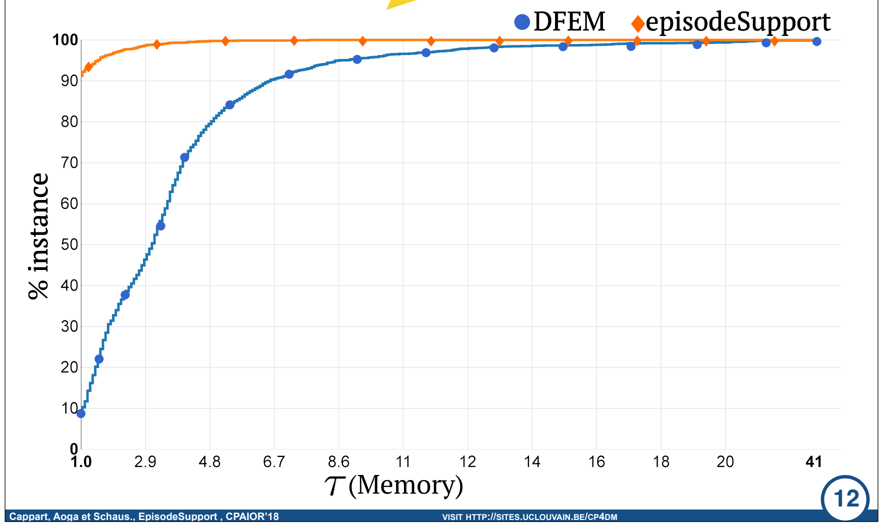
▶ EpisodeSuport (2versions — with/without time consideration)

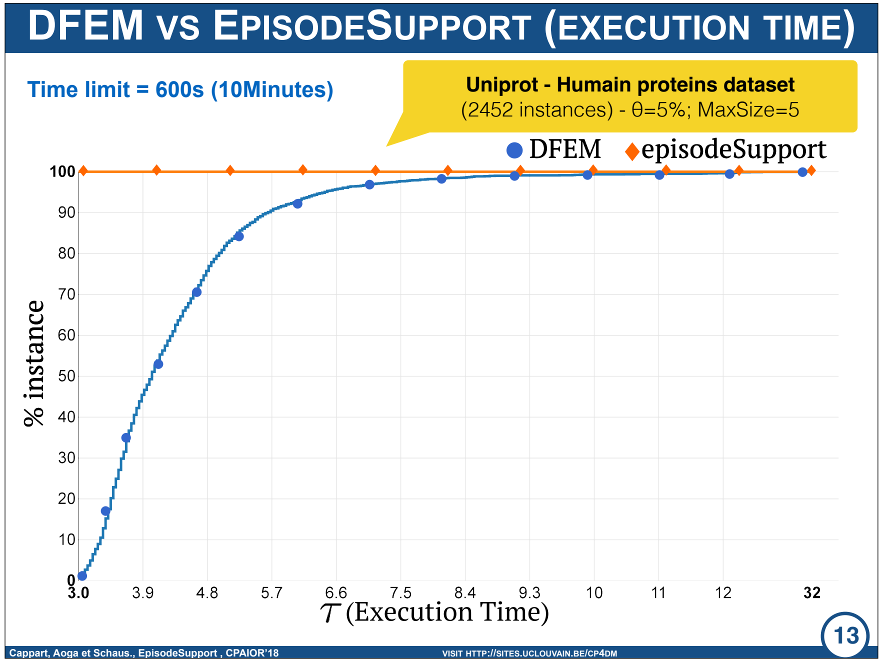
https://bitbucket.org/projetsJOHN/episodesupport (also available in OscaR)

DFEM vs EpisodeSupport (memory)

Time limit = 600s (10Minutes)

Uniprot - Humain proteins dataset
(2452 instances) - θ=5%; MaxSize=5

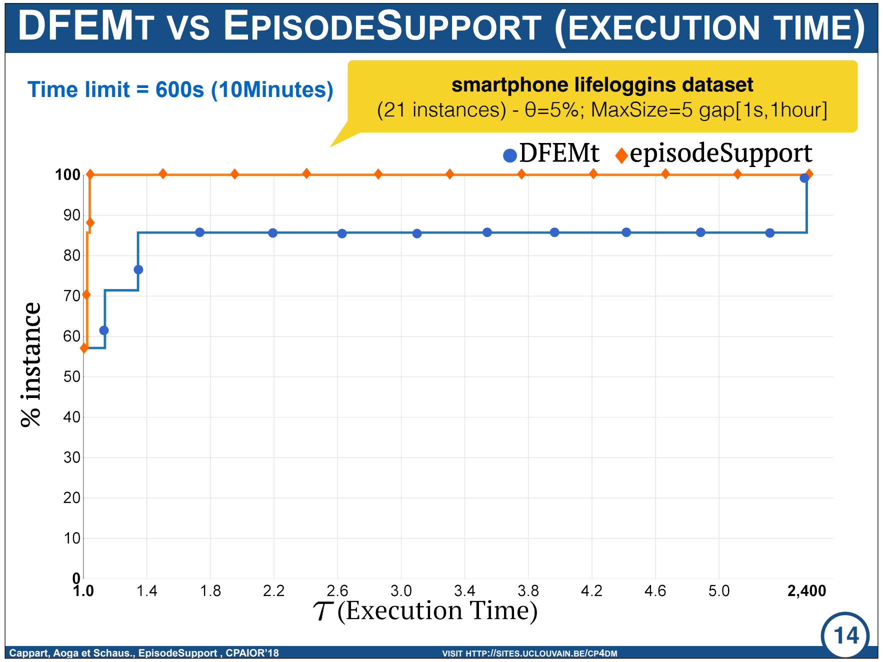




DFEM VS EPISODESUPPORT (EXECUTION TIME)

Time limit = 600s (10Minutes)

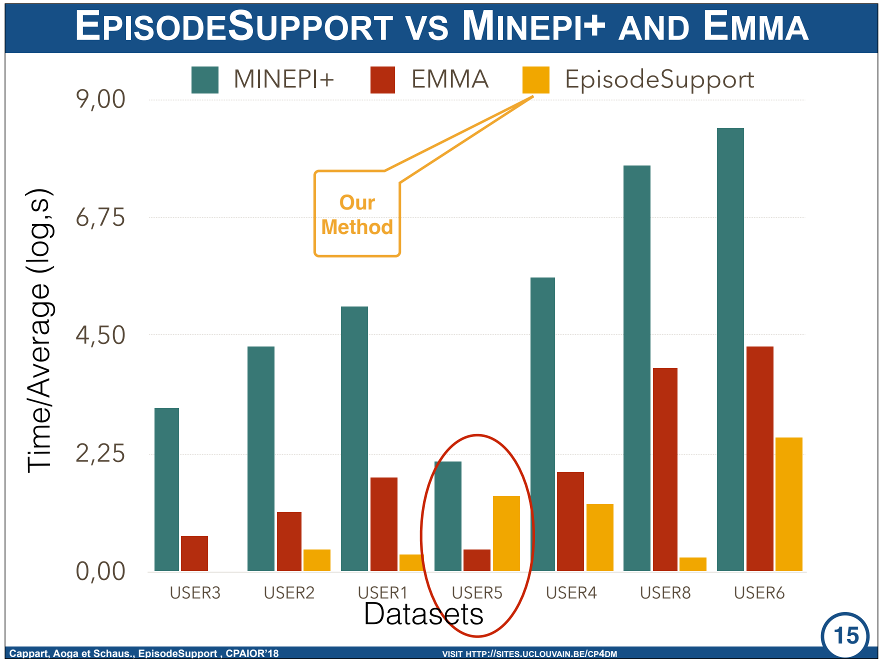
Uniprot - Humain proteins dataset
(2452 instances) - θ=5%; MaxSize=5



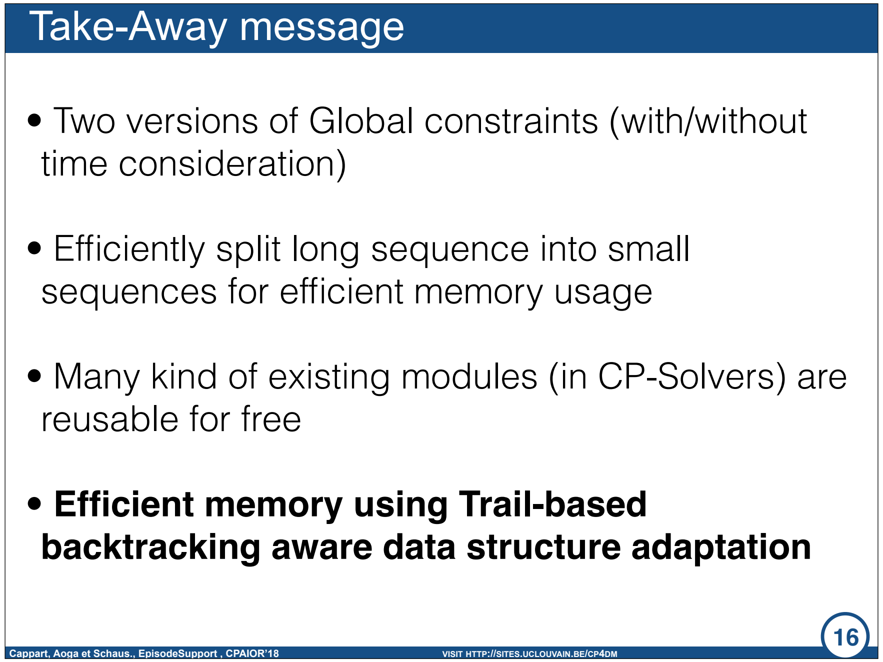
DFEMT VS EPISODESUPPORT (EXECUTION TIME)

Time limit = 600s (10Minutes)

smartphone lifeloggins dataset
(21 instances) - θ=5%; MaxSize=5 gap[1s,1hour]

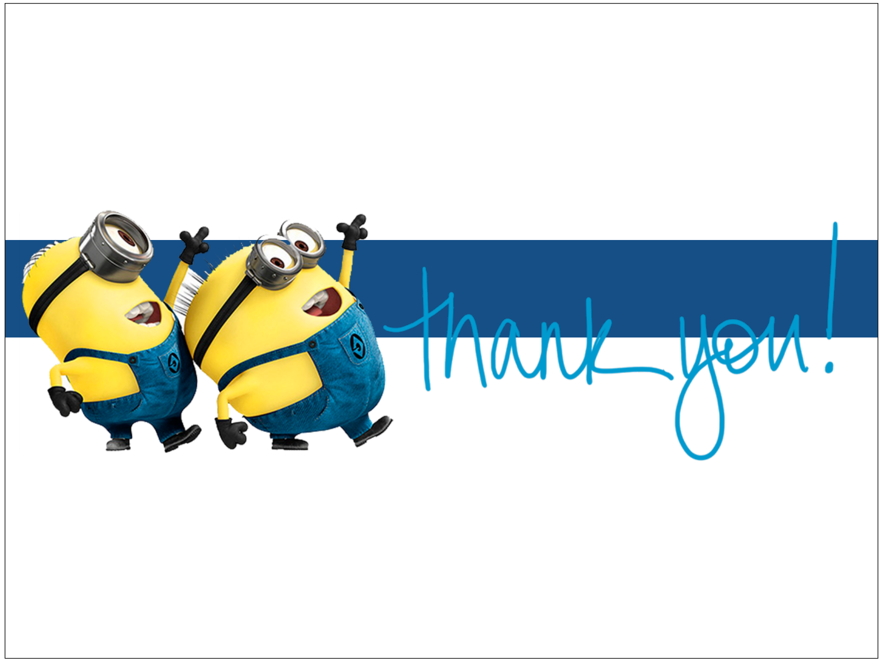


EPISODESUPPORT VS MINEPI+ AND EMMA



Take-Away message

- Two versions of Global constraints (with/without time consideration)
- Efficiently split long sequence into small sequences for efficient memory usage
- Many kind of existing modules (in CP-Solvers) are reusable for free
- **Efficient memory using Trail-based backtracking aware data structure adaptation**


thank you!