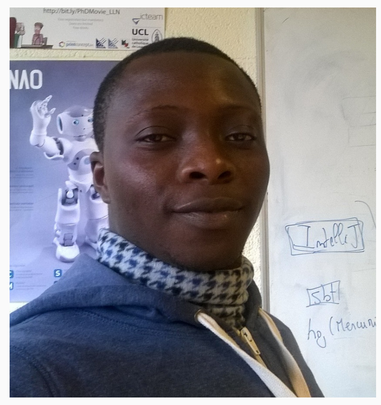
Scalable CP approach for Mining Frequent Sequence with gap constraints

http://sites.uclouvain.be/cp4dm/spm/

John O.R. Aoga(✉)[1], Pierre Schaus[1], Tias Guns[2]

[1]UCLouvain, ICTEAM, Belgium — [2]KU Leuven, DTAI Research group, Belgium

john.aoga@uclouvain.be



Abstract

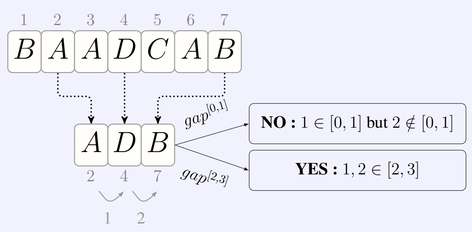
Sequence mining is an important tool for analyzing large databases of timed events, such as in click stream mining and event log mining. Recently, constraint programming (CP) approaches for pattern mining are gaining interest, due to the modularity of the framework and flexibility to add additional constraints. While CP systems were less scalable than specialized mining systems, we recently showed this can be overcome by hybridizing advanced CP techniques (trailing) with algorithmic improvements. In this work, we study the more involved task of mining under the restriction that the time *gap* between two matching events must be smaller then a threshold. We show that this too can benefit greatly from hybridization.

Problem of SPM under $gap^{[M,N]}$

Find all patterns $p = \langle p_1, p_2, \cdots, p_l \rangle$ such that at least θ sequences are matched by the pattern satisfying the gap constraints. A sequence $S \in SDB$ is matched by p `iff` $\exists (e_1, e_2, \cdots, e_l)$ such that:

- $S[e_i] = p_i \wedge i \in [1 \dots l]$
- $M \leq e_{i-1} - e_i - 1 \leq N \wedge i \in [2 \dots l]$

e_i is thus the matching position of item p_i in a sequence S.



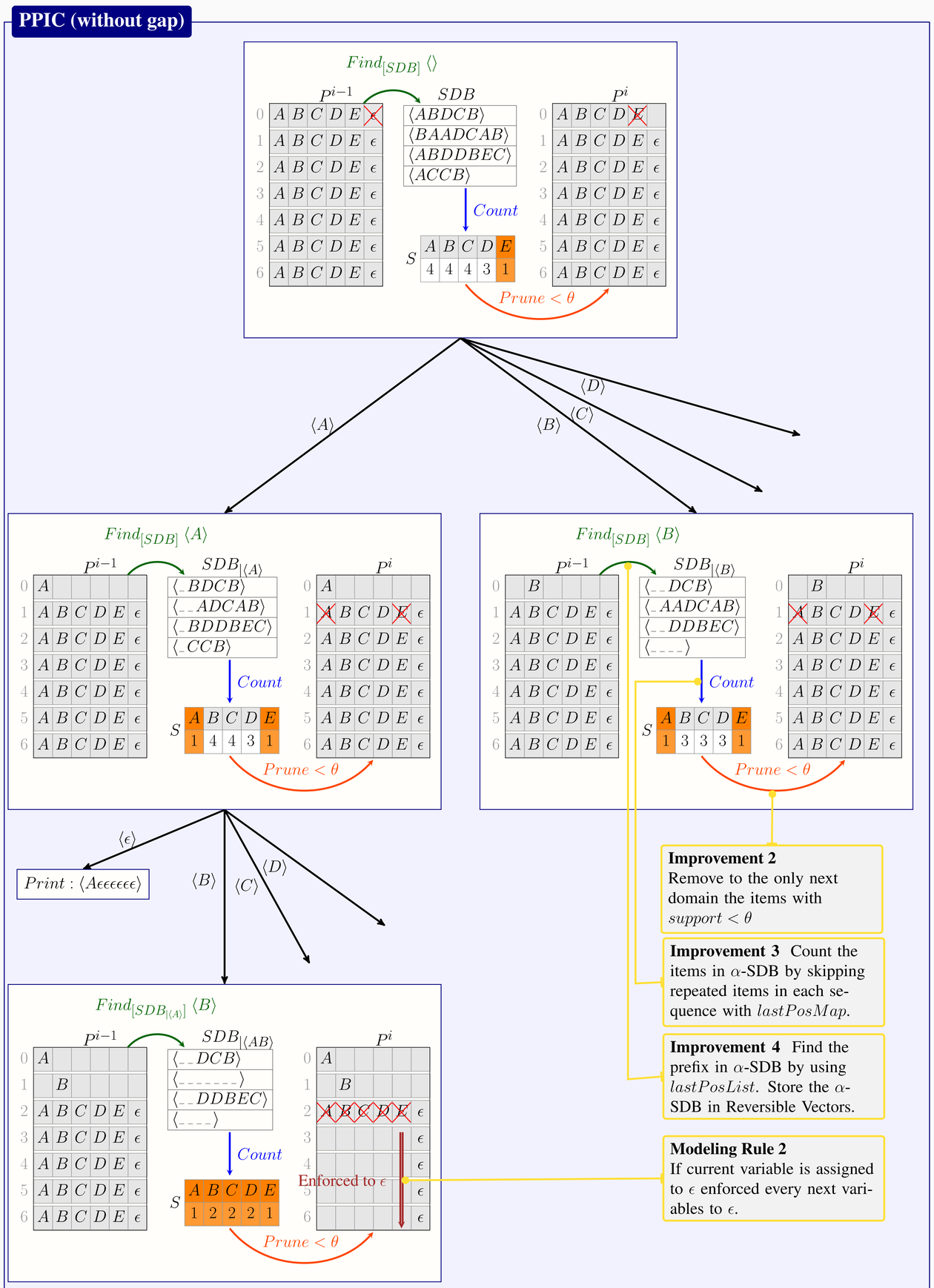
Objective

The main objective of this project is to design new constraint in CP to improve the literature.

Sample of SDB

sid	sequence	$lastPosList$	$lastPosMap$
sid_1	$\langle ABDCB \rangle$	[(B,5),(C,4),(D,3),(A,1)]	{A→1, B→5, C→4,D→3,E→0}
sid_2	$\langle BAADCAB \rangle$	[(B,7),(A,6),(C,5),(D,4)]	{A→6, B→7, C→5,D→4,E→0}
sid_3	$\langle ABDDBEC \rangle$	[(C,7),(E,6),(B,5),(D,4),(A,1)]	{A→1, B→5, C→7,D→4,E→6}
sid_4	$\langle ACCB \rangle$	[(B,4),(C,3),(A,1)]	{A→1, B→4, C→3,D→0,E→0}

PPIC (without gap)



Improvement 2 Remove to the only next domain the items with $support < \theta$

Improvement 3 Count the items in α-SDB by skipping repeated items in each sequence with $lastPosMap$.

Improvement 4 Find the prefix in α-SDB by using $lastPosList$. Store the α-SDB in Reversible Vectors.

Modeling Rule 2 If current variable is assigned to ϵ enforced every next variables to ϵ.

CP Model

A constraint model consists of variables, domains and constraints.
A CP model over $P = [P_1, P_2, \dots, P_L]$ (P_i is integer variables) represents the frequent sequence pattern with threshold θ, iff the following three conditions are satisfied by every valid assignment to P:

1. $P_1 \neq \epsilon$ (ϵ represents empty character and the end of pattern)
2. $\forall i \in \{2, \dots, L-1\} : P_i = \epsilon \Rightarrow P_{i+1} = \epsilon$
3. $\#\{(sid, s) \in SDB \ \langle P_1 \dots P_j \rangle \preceq s\} \geq \theta, j = \max(\{i \in \{1 \dots L\} | P_i \neq 0\})$.

Related Work

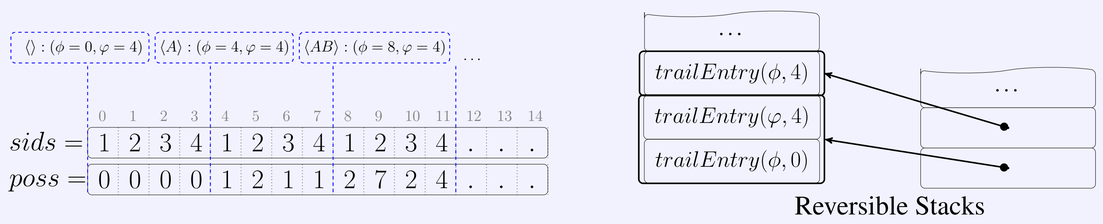
Specialised Methods :

- `PrefixSpan [6]` : prefix and prefix-projection method with DFS;
- `cSPADE [8]` : vertical database with join rules in DFS/BFS;
- `LAPIN-SPAM [7]` : idea of last position of items;

CP-based Methods :

- `CPSM [4]` : one constraint per sequence + reified constraints;
- `PP` and `GapSeq [3, 2]` : global constraint with filtering inspired of prefixSpan method + maximal gap constraint;
- `PPIC [1]` : last position of items applied in prefix-projection and Trail-based backtracking aware datastructure.

Trail-based backtracking aware datastructure (PPIC Improvement 1)

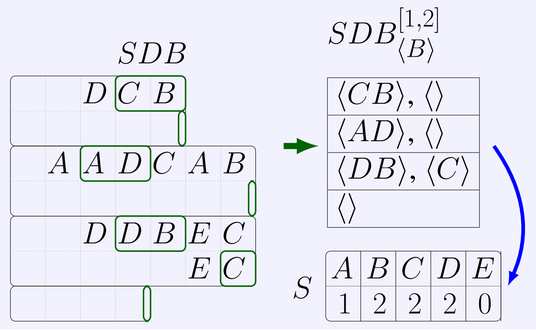


PPICgap with gap challenge

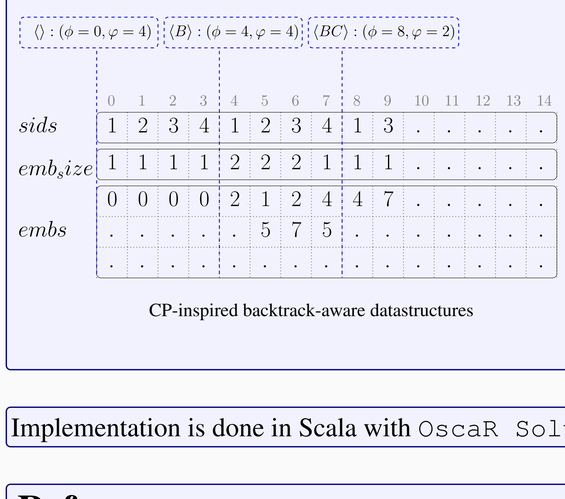
A $gap^{[M,N]}$ constraint changes when a subsequence is included in a sequence, namely iff the *gap* between two subsequent symbols $\geq M$ and $\leq N$.

`PPICgap compared to PPIC`

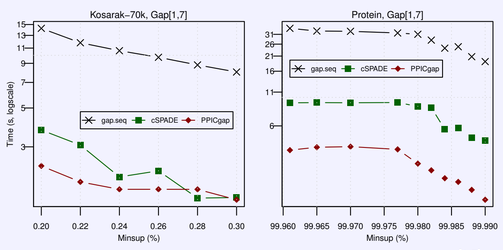
1. **Projected database**, keep all possible suffixes for each prefix;
2. **Trail-Based backtracking aware** datastructure;
3. **Pruning** is based on prefix-antimonotonicity property.



CP-inspired backtrack-aware datastructures (PPICgap)



CP-inspired backtrack-aware datastructures



CPU times for Methods for several minsup under $gap^{[1,7]}$

Implementation is done in Scala with `OscaR Solver [5]`.

References

[1] Aoga, J.O., Guns, T., Schaus, P.: An efficient algorithm for mining frequent sequence with constraint programming. LNAI,Part II, ECML PKDD 9853 (2016)

[2] Kemmar, A., Loudni, S., Lebbah, Y., Boizumault, P., Charnois, T.: A global constraint for mining sequential patterns with gap constraint. CPAIOR16 (2015)

[3] Kemmar, A., Loudni, S., Lebbah, Y., Boizumault, P., Charnois, T.: Prefix-projection global constraint for sequential pattern mining. In: Principles and Practice of Constraint Programming. Springer (2015)

[4] Negrevergne, B., Guns, T.: Constraint-based sequence mining using constraint programming. In: CPAIOR15. Springer (2015)

[5] OscaR Team: OscaR: Scala in OR (2012), available from `https://bitbucket.org/oscarlib/oscar`

[6] Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: icccn. p. 0215. IEEE (2001)

[7] Yang, Z., Kitsuregawa, M.: LAPIN-SPAM: An Improved Algorithm for Mining Sequential Pattern. In: International Conference on Data Engineering (2005)

[8] Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. Machine learning 42(1-2), 31–60 (2001)

BENELEARN 2016

BELGIAN-DUTCH CONFERENCE ON MACHINE LEARNING

https://www.kuleuven-kulak.be/benelearn/