An Efficient Algorithm for SPM with CP

J. AOGA¹, T. Guns², P. Schaus³
¹UCLouvain;²KULeuven — Belgium

Problem of Sequential Pattern Mining (SPM)

- Sequence Database (SDB)
- Sequence : < Milk Coffee Sugar Coffee Sugar>

Client 1:
Milk Coffee Sugar Coffee Sugar

Client 2:
Coffee Milk Coffee Sugar

Client 3:
Milk Coffee

Client 4:
Coffee Sugar Egg
• Sequence Database (SDB)
• Sequence : < Milk Coffee Sugar Coffee Sugar>
• Subsequence : <Coffee Sugar>

Problem : Find all subsequences with support ≥ Given Threshold
Related Work

![Timeline Diagram]

**CP: Filtering + DFSearch**

<table>
<thead>
<tr>
<th>Variables</th>
<th>P₁</th>
<th>P₂</th>
<th>P₃</th>
<th>P₄</th>
<th>P₅</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domains</strong></td>
<td>Coffee</td>
<td>Milk</td>
<td>Coffee</td>
<td>Milk</td>
<td>Coffee</td>
</tr>
<tr>
<td></td>
<td>Sugar</td>
<td>Sugar</td>
<td>Sugar</td>
<td>Sugar</td>
<td>Sugar</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

![Variables Table]

<table>
<thead>
<tr>
<th>Variables</th>
<th>P₁</th>
<th>P₂</th>
<th>P₃</th>
<th>P₄</th>
<th>P₅</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domains</strong></td>
<td>Coffee</td>
<td>Milk</td>
<td>Coffee</td>
<td>Milk</td>
<td>Coffee</td>
</tr>
<tr>
<td></td>
<td>Sugar</td>
<td>Sugar</td>
<td>Sugar</td>
<td>Sugar</td>
<td>Sugar</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

Aoga et al., An Efficient Algorithm for SPM with CP, ECML PKDD 2016

---

Related Work

![Timeline Diagram]

**CP: Filtering + DFSearch**

<table>
<thead>
<tr>
<th>Variables</th>
<th>P₁</th>
<th>P₂</th>
<th>P₃</th>
<th>P₄</th>
<th>P₅</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domains</strong></td>
<td>Coffee</td>
<td>Milk</td>
<td>Coffee</td>
<td>Milk</td>
<td>Coffee</td>
</tr>
<tr>
<td></td>
<td>Sugar</td>
<td>Sugar</td>
<td>Sugar</td>
<td>Sugar</td>
<td>Sugar</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

![Variables Table]

<table>
<thead>
<tr>
<th>Variables</th>
<th>P₁</th>
<th>P₂</th>
<th>P₃</th>
<th>P₄</th>
<th>P₅</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domains</strong></td>
<td>Coffee</td>
<td>Milk</td>
<td>Coffee</td>
<td>Milk</td>
<td>Coffee</td>
</tr>
<tr>
<td></td>
<td>Sugar</td>
<td>Sugar</td>
<td>Sugar</td>
<td>Sugar</td>
<td>Sugar</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

Aoga et al., An Efficient Algorithm for SPM with CP, ECML PKDD 2016
1. Memory and DFS improvement. How to Store and restore databases in the DFSearch? => reversible vectors making use of trailing techniques.
<table>
<thead>
<tr>
<th>Client1</th>
<th>Milk</th>
<th>Coffee</th>
<th>Sugar</th>
<th>Coffee</th>
<th>Sugar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client2</td>
<td>Coffee</td>
<td>Milk</td>
<td>Coffee</td>
<td>Sugar</td>
<td></td>
</tr>
<tr>
<td>Client3</td>
<td>Milk</td>
<td>Coffee</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client4</td>
<td>Coffee</td>
<td>Sugar</td>
<td>Egg</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Supports**

\[
\begin{array}{cccccc}
P_1 & P_2 & P_3 & P_4 & P_5 \\
Milk & Milk & Milk & Milk & Milk \\
Coffee & Coffee & Coffee & Coffee & Coffee \\
Sugar & Sugar & Sugar & Sugar & Sugar \\
Egg & Egg & Egg & Egg & Egg \\
\end{array}
\]
Given Threshold = 3 (75%)
Given Threshold=3 (75%)

Supports

- M: 3
- C: 4
- S: 3

Seq. Pos. 0 1 2 3 4
start=0 Size=4

---

Given Threshold=3 (75%)

Supports

- M: 3
- C: 4
- S: 3

Seq. Pos. 0 1 2 3 4
start=0 Size=4

---

Given Threshold=3 (75%)

Supports

- M: 3
- C: 4
- S: 3

Seq. Pos. 0 1 2 3 4
start=0 Size=4

---

Given Threshold=3 (75%)

Supports

- M: 3
- C: 4
- S: 3

Seq. Pos. 0 1 2 3 4
start=0 Size=4
Given Threshold=3 (75%)

Supports

M : 3  S : 3  C : 4

\[ P_1 \ P_2 \ P_3 \ P_4 \ P_5 \]
\[ \text{Milk Coffee Coffee Coffee Coffee} \]
\[ \text{Sugar Sugar Sugar Sugar Sugar} \]
\[ \text{Egg Egg Egg Egg Egg} \]

Top of the sub-stack

Start=0, size=4

TrailStack
Given Threshold=3 (75%)

Supports
M: 0
C: 3
S: 2
E: 4

Threshold

E : 1
S : 3
C : 4
M

\[ \text{Seq.} \quad \text{Pos.} \quad 1 \quad 0 \quad 3 \quad 0 \quad 4 \quad 0 \quad 2 \quad 0 \quad 1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 1 \]

Given:
Milk Coffee Sugar Egg

TrailStack

Start=0, size=4

Top of the sub-stack

Seq.
1
2
3
0

Pos.
1
2
1
0

Start=4, Size=3

Supports
M: 3
C: 1
S: 2
E: 4

Threshold

E : 1
S : 3
C : 4
M

\[ \text{Seq.} \quad \text{Pos.} \quad 1 \quad 0 \quad 3 \quad 0 \quad 4 \quad 0 \quad 2 \quad 0 \quad 1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 1 \]

Given:
Milk Coffee Sugar Egg

TrailStack

Start=0, size=4

Top of the sub-stack

Seq.
1
2
3
0

Pos.
1
2
1
0

Start=4, Size=3

Supports
M: 3
C: 1
S: 2
E: 4

Threshold

E : 1
S : 3
C : 4
M

\[ \text{Seq.} \quad \text{Pos.} \quad 1 \quad 0 \quad 3 \quad 0 \quad 4 \quad 0 \quad 2 \quad 0 \quad 1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 1 \]

Given:
Milk Coffee Sugar Egg

TrailStack

Start=0, size=4

Top of the sub-stack

Seq.
1
2
3
0

Pos.
1
2
1
0

Start=4, Size=3

Supports
M: 3
C: 1
S: 2
E: 4

Threshold

E : 1
S : 3
C : 4
M

\[ \text{Seq.} \quad \text{Pos.} \quad 1 \quad 0 \quad 3 \quad 0 \quad 4 \quad 0 \quad 2 \quad 0 \quad 1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 1 \]

Given:
Milk Coffee Sugar Egg

TrailStack

Start=0, size=4

Top of the sub-stack

Seq.
1
2
3
0

Pos.
1
2
1
0

Start=4, Size=3

Supports
M: 3
C: 1
S: 2
E: 4

Threshold

E : 1
S : 3
C : 4
M

\[ \text{Seq.} \quad \text{Pos.} \quad 1 \quad 0 \quad 3 \quad 0 \quad 4 \quad 0 \quad 2 \quad 0 \quad 1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 1 \]

Given:
Milk Coffee Sugar Egg

TrailStack

Start=0, size=4

Top of the sub-stack

Seq.
1
2
3
0

Pos.
1
2
1
0

Start=4, Size=3

Supports
M: 3
C: 1
S: 2
E: 4

Threshold

E : 1
S : 3
C : 4
M

\[ \text{Seq.} \quad \text{Pos.} \quad 1 \quad 0 \quad 3 \quad 0 \quad 4 \quad 0 \quad 2 \quad 0 \quad 1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 1 \]

Given:
Milk Coffee Sugar Egg

TrailStack

Start=0, size=4

Top of the sub-stack

Seq.
1
2
3
0

Pos.
1
2
1
0

Start=4, Size=3

Supports
M: 3
C: 1
S: 2
E: 4

Threshold

E : 1
S : 3
C : 4
M

\[ \text{Seq.} \quad \text{Pos.} \quad 1 \quad 0 \quad 3 \quad 0 \quad 4 \quad 0 \quad 2 \quad 0 \quad 1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 1 \]

Given:
Milk Coffee Sugar Egg

TrailStack

Start=0, size=4

Top of the sub-stack

Seq.
1
2
3
0

Pos.
1
2
1
0

Start=4, Size=3
Given Threshold = 3 (75%)

Given Threshold = 3 (75%)

Given Threshold = 3 (75%)

Given Threshold = 3 (75%)
1. Memory and DFS improvement. How to Store and restore databases in the DFSearch? => reversible vectors making use of trailing techniques.

2. Support Count Improvement. How to compute support efficiently? Visit only the last position of each symbol after start position.

How to compute items support in SDB?
- Identify items which exist in each sequence of SDB and increase items support value.
- Often many repeating symbols: cache for each symbol its ‘last position’ and only iterate over those (O(m) vs O(n))

- startPos=5

△ Last Position List = [(B,16),(C,12),(A,11),(D,2),(E,0)]

Aoga et al., An Efficient Algorithm for SPM with CP, ECML PKDD 2016

9
### Improvements of Literature (3/4)

1. **Memory and DFS improvement.** How to store and restore databases in the DFSearch? => reversible vectors making use of trailing techniques.

2. **Support Count Improvement.** How to compute items support efficiently? Visit only the last position of each symbol after start position.

3. **Sequence projection Improvement.** $P_1P_2P_3P_4P_5$ has just been extended to $E P_2P_3P_4P_5$. **Can we decide in $O(1)$ if an embedding can not be extended?**

---

**Naive Solution:**
Scan all items in each sequence. $O(n)$ check per sequence.

**Can we do better? YES**
For each sequence, precompute a map from symbol to last position in the sequence. If sequence is extended by 'E', verify $\text{lastPos}[\text{sid}][E]$ is larger than startPos. $O(1)$ check per sequence.
1. Which sequences contain at least one E?

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>C</td>
<td>S</td>
<td>C</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>M</td>
<td>C</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>M</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>S</td>
<td>E</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Naive Solution:** Scan all Items in each sequence. O(n) check per sequence.

Can we do better? YES
For each sequence, precompute a map from symbol to last position in the sequence. If sequence is extended by 'E', verify lastPos[sid][E] is larger than startPos. O(1) check per sequence.

0 > 0
Improvements of Literature (4/4)

1. Memory and DFS improvement. How to Store and restore databases in the DFSearch? => reversible vectors making use of trailing techniques.

2. Support Count Improvement. How to compute items support efficiently? Visit only the last position of each symbol after start position.

3. Sequence visited Improvement. P₁P₂P₃P₄P₅ has just been extended to EP₂P₃P₄P₅. Can we decide in O(1) if an embedding can not be extended?

4. Pruning Improvement. Remove infrequent item from only Dᵢ₊₁ domains of Pᵢ₊₁.

Results

Three Global constraints in OscaR implemented with Scala

- **PPIC**: memory + support counting + sequence visited + prune improvements
- **PPDC**: same with PPIC but support counting is different.
- **PPmixed**: Mixed of PPIC/PPDC base on heuristic

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

Compared with CP-Based methods

Aoga et al., An Efficient Algorithm for SPM with CP, ECML PKDD 2016

Aoga et al., An Efficient Algorithm for SPM with CP, ECML PKDD 2016
Compared with Specialized methods

Handling of different additional constraints

Conclusion

- Integrate ideas of both SPM and CP in a global constraint for CP solver (Generic and Flexible);
- First time, CP-based approach outperforms both CP-based approaches and Specialized methods;
- Efficient memory using Trail-based backtracking aware datastructure really speed up search in DFS search;
- Compatible with a number of other constraints, such as: minimum and maximum length constraint, Regular expression constraint, etc
- Future work: use generality of framework for more constraints and other sequence mining settings (multi-objective, information theory based, interactive, ...)
- Code and apps are open http://sites.uclouvain.be/cp4dm/spm/

thank you!
1. Memory and DFS improvement. How to Store and restore databases in the DFSearch? => reversible vectors making use of trailing techniques.
<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>C</th>
<th>S</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Supports: M: 4, C: 3, S: 3, E: 1

Given Threshold = 3 (75%)
Given Threshold=3 (75%)
Given Threshold = 3 (75%)

Supports
M : 3
C : 4
S : 3

P₁ P₂ P₃ P₄ P₅
Milk Coffee Coffee Coffee Coffee
Sugar Sugar Sugar Sugar Sugar
Egg Egg Egg Egg Egg

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

Given Threshold = 3 (75%)

Supports
M : 3
C : 4
S : 3

P₁ P₂ P₃ P₄ P₅
Milk Coffee Coffee Coffee Coffee
Sugar Sugar Sugar Sugar Sugar
Egg Egg Egg Egg Egg

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

Given Threshold = 3 (75%)

Supports
M : 3
C : 4
S : 3

P₁ P₂ P₃ P₄ P₅
Milk Coffee Coffee Coffee Coffee
Sugar Sugar Sugar Sugar Sugar
Egg Egg Egg Egg Egg

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

Given Threshold = 3 (75%)

Supports
M : 3
C : 4
S : 3

P₁ P₂ P₃ P₄ P₅
Milk Coffee Coffee Coffee Coffee
Sugar Sugar Sugar Sugar Sugar
Egg Egg Egg Egg Egg

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

start=4
Size=3

start=4
Size=3

start=0
Size=4

start=0
Size=4

21

21

21

21
Given Threshold=3 (75%)

Supports
M: 3
C: 4
S: 3

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

Top of the sub-stack
TrailStack
start=0, size=4

Given Threshold=3 (75%)

Supports
M: 3
C: 4
S: 3

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

Top of the sub-stack
TrailStack
start=0, size=4
Given Threshold=3 (75%) Given Threshold=3 (75%) Given Threshold=3 (75%) Given Threshold=3 (75%)

 Supports M: 3 C: 4 S: 3 Supports M: 3 C: 4 S: 3 Supports M: 3 C: 4 S: 3 Supports M: 3 C: 4 S: 3

\[
\begin{align*}
S &= \text{Supports} \\
M &= \text{Given} \\
S &= \text{Threshold} = 3 \\
(75\%)
\end{align*}
\]

\[
\begin{align*}
0 & 1 2 3 4 \\
M & C S C S \\
\text{Start} & = 0, \text{Size} = 4
\end{align*}
\]

\[
\begin{align*}
\text{Supports} &\leftarrow M: 3 \\
\text{Milk} &\leftarrow P_1 \\
\text{Coffee} &\leftarrow P_2 \\
\text{Sugar} &\leftarrow P_3 \\
\text{Egg} &\leftarrow P_4 \\
\text{Seq.} &\leftarrow 1 \\
\text{Pos.} &\leftarrow 0 \\
\text{TrailStack} &\leftarrow \text{Top of the sub-stack}
\end{align*}
\]
Improvements of Literature (2/4)

1. Memory and DFS improvement. How to Store and restore databases in the DFSearch? => reversible vectors making use of trailing techniques.

2. Support Count Improvement. How to compute support efficiently? Visit only the last position of each symbol after start position.

How to compute items support in SDB?

- Identify items which exist in each sequence of SDB and increase items support value.

- Often many repeating symbols: cache for each symbol its 'last position' and only iterate over those (O(m) vs O(n))

- startPos=5

How to compute items support in SDB?

- Identify items which exist in each sequence of SDB and increase items support value.

- Often many repeating symbols: cache for each symbol its 'last position' and only iterate over those (O(m) vs O(n))

- startPos=5

Last Position List = [(B,16),(C,12),(A,11),(D,2),(E,0)]
How to compute items support in SDB?

- Identify items which exist in each sequence of SDB and increase items support value.

- Often many repeating symbols: cache for each symbol its 'last position' and only iterate over those (\(O(m)\) vs \(O(n)\))

- startPos=5

Last Position List = \([(B,16),(C,12),(A,11),(D,2),(E,0)]\)

Aoga et al., An Efficient Algorithm for SPM with CP, ECML PKDD 2016

Improvements of Literature (3/4)

1. Memory and DFS improvement. How to Store and restore databases in the DFS? \(\Rightarrow\) reversible vectors making use of trailing techniques.

2. Support Count Improvement. How to compute items support efficiently? Visit only the last position of each symbol after startPos.

3. Sequence projection Improvement. \(P_1P_2P_3P_4P_5\) has just been extended to \(EP_2P_3P_4P_5\). Can we decide in \(O(1)\) if an embedding can not be extended?

1. Which sequences contain at least one \(E\)?

Naive Solution: Scan all Items in each sequence. \(O(n)\) check per sequence
1. Which sequences contain at least one $E$?

**Naive Solution:** Scan all items in each sequence. O(n) check per sequence

Can we do better? YES
For each sequence, precompute a map from symbol to last position in the sequence. If sequence is extended by 'E', verify $\text{lastPos}[\text{sid}][E]$ is larger than $\text{startPos}$. O(1) check per sequence

---

1. Which sequences contain at least one $E$?

**Naive Solution:** Scan all items in each sequence. O(n) check per sequence

Can we do better? YES
For each sequence, precompute a map from symbol to last position in the sequence. If sequence is extended by 'E', verify $\text{lastPos}[\text{sid}][E]$ is larger than $\text{startPos}$. O(1) check per sequence

---

1. Which sequences contain at least one $E$?

**Naive Solution:** Scan all items in each sequence. O(n) check per sequence

Can we do better? YES
For each sequence, precompute a map from symbol to last position in the sequence. If sequence is extended by 'E', verify $\text{lastPos}[\text{sid}][E]$ is larger than $\text{startPos}$. O(1) check per sequence

---

1. Which sequences contain at least one $E$?

**Naive Solution:** Scan all items in each sequence. O(n) check per sequence

Can we do better? YES
For each sequence, precompute a map from symbol to last position in the sequence. If sequence is extended by 'E', verify $\text{lastPos}[\text{sid}][E]$ is larger than $\text{startPos}$. O(1) check per sequence
1. Which sequences contain at least one E?

Naive Solution: Scan all items in each sequence. O(n) check per sequence.

Can we do better? YES. For each sequence, precompute a map from symbol to last position in the sequence. If sequence is extended by E, verify lastPos[sid][E] is larger than startPos. O(1) check per sequence.

Improvements of Literature (4/4)

1. Memory and DFS improvement. How to Store and restore databases in the DFSearch? => reversible vectors making use of trailing techniques.

2. Support Count Improvement. How to compute items support efficiently? Visit only the last position of each symbol after start position.

3. Sequence visited Improvement. P₁P₂P₃P₄P₅ has just been extended to EP₂P₃P₄P₅. Can we decide in O(1) if an embedding can not be extended?

4. Pruning Improvement. Remove infrequent item from only D₁,1 domains of P₁,1.

Experiments

Results

Three Global constraints in OscaR implemented with Scala

- PPIC: memory + support counting + sequence visited + prune improvements
- PPDC: same with PPIC but support counting is different.
- PPmixed: Mixed of PPIC/PPDC base on heuristic

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

Aoga et al., An Efficient Algorithm for SPM with CP. ECML PKDD 2016
Compared with CP-Based methods

- PPIC
- GapSeq
- cSPADE

Handling of different additional constraints

- Minimum and maximum pattern size constraint
- Item constraint
- Regular expression constraint

Conclusion

- Integrate ideas of both SPM and CP in a global constraint for CP solver (Generic and Flexible);
- First time, CP-based approach outperforms both CP-based approaches and Specialized methods;
- Efficient memory using Trail-based backtracking aware datastructure really speed up search in DFSearch;
- Compatible with a number of other constraints, such as: minimum and maximum length constraint, Regular expression constraint, etc
- Future work: use generality of framework for more constraints and other sequence mining settings (multi-objective, information theory based, interactive, ...)
- Code and apps are open http://sites.uclouvain.be/cp4dm/spm/
thank you!