Debugging Embedded Multimedia Application
Execution Traces through Periodic Pattern Mining

Patricia López Cueva

CIFRE Thesis supervised by:
Alexandre Termier, Jean François Méhaut and Miguel Santana

8th July, 2013
STMicroelectronics
Leading semiconductor manufacturer.

Telecommunications and Multimedia
- Highly integrated devices.
- Competitive market: Hardware + Software.
- Software development: difficult and slow.
- Time-to-market.
- Debugging phase: long and costly.
Debugging Techniques

Software Debugging

Functional Debugging
Interactive debuggers ⇒ High Intrusiveness

Performance Debugging
Profiler ⇒ Not enough detail

More Parallelism

- More bugs: interaction between components of the system.
- Interactive debuggers or profilers not suitable to diagnose these bugs.

Tracing: A multipurpose solution

- Recording the execution of the application for a postmortem analysis.
- Problem: Their size is becoming unmanageable for a manual analysis.
- Need: Automatic analysis tools.

Patricia López Cueva
Thesis Defense
8th July, 2013
Pattern Mining: A possible solution

Data Mining
Extract knowledge from huge volumes of data.

Frequent Pattern Mining
Discover regularities in the data that are called patterns.
Fine-grained analysis of the application behavior.

Characteristic of multimedia application
Periodic behavior (frame decoding).

Proposal
A new approach to debug multimedia application execution traces through periodic pattern mining.
Outline

1 Introduction
   - Context and Motivation

2 Formal Framework
   - Frequent Periodic Patterns
   - Core Periodic Concepts
   - PerMiner Algorithm
   - Scalability Experiments

3 Its Application
   - Usage Guidelines
   - Use Cases

4 Conclusions
   - Conclusions
   - Future Work
Outline

1. Introduction
   - Context and Motivation

2. Formal Framework
   - Frequent Periodic Patterns
   - Core Periodic Concepts
   - PerMiner Algorithm
   - Scalability Experiments

3. Its Application
   - Usage Guidelines
   - Use Cases

4. Conclusions
   - Conclusions
   - Future Work
What is a periodic pattern?

**Periodic Pattern**: Group of events that appear regularly in the trace.

**Pattern Mining**: Set of items that appear periodically in the transactional DB.

---

**Execution Trace (s.\(\mu\)s)**

- 68.770630 getFrame
- 68.770697 displayFrame
- 68.770741 int16
- 68.770768 swint16
- 68.770869 displayFrame
- 68.770913 getFrame
- 68.770959 write16
- 68.770982 cpu_clock
- 68.771099 displayFrame
- 68.771150 read16
- 68.771235 fork
- 68.771324 getpid
- 68.771346 getFrame
- 68.771372 displayFrame
- 68.771402 printk
- 68.771456 sem_up
- 68.771487 sem_down
- 68.771540 getFrame
- 68.771586 displayFrame
What is a periodic pattern?

**Periodic Pattern**

Group of events that appear regularly in the trace.

**Pattern Mining**

Set of items that appear periodically in the transactional DB.

**Execution Trace (s.µs)**

- 68.770630 getFrame
- 68.770697 displayFrame
- 68.770741 int16
- 68.770768 swint16
- 68.770869 displayFrame
- 68.770913 getFrame
- 68.770959 write16
- 68.770982 cpu_clock
- 68.771032 getFrame
- 68.771099 displayFrame
- 68.771150 read16
- 68.771235 fork
- 68.771324 get_pid
- 68.771346 getFrame
- 68.771372 displayFrame
- 68.771402 printk
- 68.771456 sem_up
- 68.771487 sem_down
- 68.771540 getFrame
- 68.771586 displayFrame
What is a periodic pattern?

Periodic Pattern: Group of events that appear regularly in the trace.

Pattern Mining: Set of items that appear periodically in the transactional DB.

Execution Trace (s.\(\mu\)s)

Preprocessing

<table>
<thead>
<tr>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
</tr>
<tr>
<td>t2</td>
</tr>
<tr>
<td>t3</td>
</tr>
<tr>
<td>t4</td>
</tr>
<tr>
<td>t5</td>
</tr>
<tr>
<td>t6</td>
</tr>
<tr>
<td>t7</td>
</tr>
<tr>
<td>t8</td>
</tr>
<tr>
<td>t9</td>
</tr>
<tr>
<td>t10</td>
</tr>
</tbody>
</table>
Cycle \((\text{itemset}, \text{period}, \text{offset}, \text{length})\)

<table>
<thead>
<tr>
<th>t_1</th>
<th>t_2</th>
<th>t_3</th>
<th>t_4</th>
<th>t_5</th>
<th>t_6</th>
<th>t_7</th>
<th>t_8</th>
<th>t_9</th>
<th>t_{10}</th>
</tr>
</thead>
<tbody>
<tr>
<td>gF</td>
<td>dF</td>
<td>gF</td>
<td>dF</td>
<td>w16</td>
<td>clk</td>
<td>r16</td>
<td>fF</td>
<td>gpid</td>
<td>dF</td>
</tr>
<tr>
<td>SI16</td>
<td></td>
<td>dF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Cycle** \(\{\text{gF, dF}\}, 2, 1, 3\)

- **Itemset**: Set of events.
- **Period**: Distance between two consecutive transactions of the cycle.
- **Offset**: Transaction identifier of first transaction forming part of the cycle.
- **Length**: Number of transactions forming part of the cycle.

**Legend**:  
- \(gF\) getFrame, \(dF\) displayFrame, \(I16\) int16, \(SI16\) swint16, \(W16\) write16, \(clk\) cpu_clock, \(R16\) read16, \(fF\) fork, \(gpid\) get_pid, \(pk\) printk, \(sup\) sem_up, \(sd\) sem_down.
**Periodic Pattern** $P(\text{itemset}, \text{period}, \text{support}, \text{cycles})$

A group of cycles forms a periodic pattern if:

1. Same period for all cycles.
2. All cycles are consecutive.
3. Cycles do not overlap.

### Support

Sum of all cycles lengths:

$$cycles = \{(o_1, l_1), ..., (o_k, l_k)\}$$

$$support = \sum_{i=1}^{k} l_i$$

### Frequent Periodic Pattern

Given a minimum support threshold (min_sup), a pattern is frequent if

$$support \geq min\_sup$$
How many Frequent Periodic Patterns?

High redundancy!

1. All combinations of a large itemset.
2. All combinations of frequent periods.

<table>
<thead>
<tr>
<th>Frequent Periodic Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_1({gF}, 2, 5, {(1, 3)(8, 2)}))</td>
</tr>
<tr>
<td>(P_2({dF}, 2, 5, {(1, 3)(8, 2)}))</td>
</tr>
<tr>
<td>(P_3({gF, dF}, 2, 5, {(1, 3)(8, 2)}))</td>
</tr>
<tr>
<td>(P_6({gF, dF}, 3, 2, {(5, 2)}))</td>
</tr>
<tr>
<td>(P_9({gF, dF}, 4, 2, {(1, 2)}))</td>
</tr>
<tr>
<td>(P_{12}({gF, dF}, 5, 2, {(3, 2)}))</td>
</tr>
<tr>
<td>(P_{15}({gF, dF}, 5, 2, {(5, 2)}))</td>
</tr>
</tbody>
</table>
Triadic Approach [Lehmann et al., 1995]

Triadic Context

3 sets
+
1 ternary relation

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Periods</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions</td>
<td>t₁</td>
<td>t₂</td>
<td>t₃</td>
</tr>
<tr>
<td>gF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>dF</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Periods</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions</td>
<td>t₁</td>
<td>t₂</td>
<td>t₃</td>
</tr>
<tr>
<td>gF</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>dF</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Set of periodic concepts

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Periods</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Transactions</td>
<td>t₁</td>
<td>t₂</td>
</tr>
<tr>
<td>gF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>dF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Periods</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Transactions</td>
<td>t₁</td>
<td>t₂</td>
</tr>
<tr>
<td>gF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>dF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Triples**

\( (\{gF, dF\}, \{2\}, \{t₁, t₃, t₅\}) \)
Set of periodic concepts

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Periods 2</th>
<th></th>
<th>Periods 3</th>
<th></th>
<th></th>
<th>Periods 4</th>
<th></th>
<th>Periods 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Transactions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gF</td>
<td>X X X X X X</td>
<td></td>
<td>X X</td>
<td></td>
<td></td>
<td>X X</td>
<td></td>
<td>X X</td>
</tr>
<tr>
<td>dF</td>
<td>X X X X X X</td>
<td></td>
<td>X X</td>
<td></td>
<td></td>
<td>X X</td>
<td></td>
<td>X X</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Periods 4</th>
<th></th>
<th>Periods 5</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Transactions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gF</td>
<td>X X</td>
<td></td>
<td>X X</td>
<td>X X</td>
</tr>
<tr>
<td>dF</td>
<td>X X</td>
<td></td>
<td>X X</td>
<td>X X</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Periodic Concepts

\[ T_1(\{gF, dF\}, \{2\}, \{t_1, t_3, t_5, t_8, t_{10}\}) \]
Set of periodic concepts

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Periods</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Transactions</td>
<td>t₁</td>
<td>t₂</td>
</tr>
<tr>
<td>gF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>dF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Periods</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Transactions</td>
<td>t₁</td>
<td>t₂</td>
</tr>
<tr>
<td>gF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>dF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Periodic Concepts

\[
T₁(\{gF, dF\}, \{2\}, \{t₁, t₃, t₅, t₈, t₁₀\})
\]

\[
T₂(\{gF, dF\}, \{2, 4\}, \{t₁, t₅\})
\]
### Set of periodic concepts

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Periods</th>
<th>2</th>
<th>3</th>
<th></th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Transactions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>t₁</td>
<td>t₂</td>
<td>t₃</td>
<td>t₄</td>
<td>t₅</td>
<td>t₆</td>
</tr>
<tr>
<td>gF</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>dF</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Periodic Concepts**

\[
T₁(\{gF, dF\}, \{2\}, \{t₁, t₃, t₅, t₈, t₁₀\})
\]

\[
T₂(\{gF, dF\}, \{2, 4\}, \{t₁, t₅\})
\]

\[
T₃(\{gF, dF\}, \{2, 5\}, \{t₃, t₅, t₈, t₁₀\})
\]
## Set of periodic concepts

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Transactions</th>
<th>Periods</th>
<th>2</th>
<th>3</th>
<th>Periods</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>t₁</td>
<td>t₂</td>
<td>t₃</td>
<td>t₄</td>
<td>t₅</td>
</tr>
<tr>
<td>gF</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>dF</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Periodic Concepts

- $T₁(\{gF, dF\}, \{2\}, \{t₁, t₃, t₅, t₈, t₁₀\})$
- $T₂(\{gF, dF\}, \{2, 4\}, \{t₁, t₅\})$
- $T₃(\{gF, dF\}, \{2, 5\}, \{t₃, t₅, t₈, t₁₀\})$
- $T₄(\{gF, dF\}, \{2, 3, 5\}, \{t₅, t₈\})$
Core Periodic Concepts (CPC)

Core Periodic Concept

A periodic concept \((I, P, T)\) is a core periodic concept if there does not exist any other periodic concept \((I', P', T')\) such that \(I = I'\), \(P' \subset P\) and \(T' \supset T\).

Periodic Concepts

\[
\begin{align*}
T_1(\{gF, dF\}, \{2\}, \{t_1, t_3, t_5, t_8, t_{10}\}) \\
T_2(\{gF, dF\}, \{2, 4\}, \{t_1, t_5\}) \\
T_3(\{gF, dF\}, \{2, 5\}, \{t_3, t_5, t_8, t_{10}\}) \\
T_4(\{gF, dF\}, \{2, 3, 5\}, \{t_5, t_8\})
\end{align*}
\]
Core Periodic Concepts (CPC)

Core Periodic Concept

A periodic concept \((I, P, T)\) is a **core periodic concept** if there does not exist any other periodic concept \((I', P', T')\) such that \(I = I'\), \(P' \subset P\) and \(T' \supset T\).

<table>
<thead>
<tr>
<th>Core Periodic Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T_1({gF, dF}, {2}, {t_1, t_3, t_5, t_8, t_{10}}))</td>
</tr>
<tr>
<td>(T_2({gF, dF}, {2, 4}, {t_1, t_5}))</td>
</tr>
<tr>
<td>(T_3({gF, dF}, {2, 5}, {t_3, t_5, t_8, t_{10}}))</td>
</tr>
<tr>
<td>(T_4({gF, dF}, {2, 3, 5}, {t_5, t_8}))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(T_1)</th>
<th>(T_2)</th>
<th>(T_3)</th>
<th>(T_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(gF)</td>
<td>(gF)</td>
<td>(gF)</td>
<td>(gF)</td>
</tr>
<tr>
<td>(dF)</td>
<td>(dF)</td>
<td>(dF)</td>
<td>(dF)</td>
</tr>
<tr>
<td>({gF, dF})</td>
<td>({gF, dF})</td>
<td>({gF, dF})</td>
<td>({gF, dF})</td>
</tr>
<tr>
<td>({2})</td>
<td>({2, 4})</td>
<td>({2, 5})</td>
<td>({2, 3, 5})</td>
</tr>
<tr>
<td>({t_1, t_3, t_5, t_8, t_{10}})</td>
<td>({t_1, t_5})</td>
<td>({t_3, t_5, t_8, t_{10}})</td>
<td>({t_5, t_8})</td>
</tr>
</tbody>
</table>
A condensed representation of a set $S$, is a subset $C$ of the set $S$ such that every element in $S$ can be derived efficiently from $C$.

- Advantage: Less results.
- Algorithms to mine them are complex to design.
- Examples: closed [Pasquier et al., 1999], non-derivable [Calders & Goethals, 2002], etc.
How do we find CPCs?

3-STEP [Lopez Cueva et al., 2012]

1. Generate all triples.
2. Mine all periodic concepts using DATA-PEELER [Cerf et al., 2009]
3. Extract CPCs.

Is there a more efficient way?

- 3-STEP needs the whole set of periodic concepts to say whether a periodic concept is a CPC.
- Connectivity property: only needs a periodic concept.
Can we directly enumerate CPCs?

- There exist enumeration techniques to enumerate directly condensed representations [Uno et al., 2004] [Arimura & Uno, 2009].
- Polynomial delay time and polynomial space complexity.

PerMiner:

- **PerMiner** bases its enumeration on these techniques: item enumeration with a special period handling.
- Depth-first search algorithm based on LCM [Uno et al., 2004] algorithm.
- Preserves polynomial delay time and polynomial space complexity (proven).
- Proven soundness, completeness and no duplicate generation.
**PerMiner: How does it work?**

\[ \{gF\} \]

\[ gF \]

\[ \perp \]
**PERMINER: How does it work?**

\[
\begin{align*}
(g_F, 2, \{t_1, t_3, t_5, t_8, t_{10}\}) & \quad (g_F, 3, \{t_5, t_8\}) & \quad (g_F, 4, \{t_1, t_5\}) & \quad (g_F, 5, \{t_3, t_5, t_8, t_{10}\})
\end{align*}
\]

**Period Computation**

\[
\{g_F\}
\]

\[
g_F
\]
**PerMiner: How does it work?**

- **PerMiner** is a tool for finding patterns in data.

- It works by identifying frequent itemsets and their periods.

- **Period Computation** involves finding the time intervals that contain these patterns.

- The process is illustrated using a graph with nodes representing itemsets and edges showing their relationships.

- The graph includes nodes labeled with itemsets and periods, such as:
  - \((gF, 2, \{t_1, t_3, t_5, t_8, t_{10}\})\)
  - \((gF, 3, \{t_5, t_8\})\)
  - \((gF, 4, \{t_1, t_5\})\)
  - \((gF, 5, \{t_3, t_5, t_8, t_{10}\})\)

- These itemsets are connected through a recursive call, indicating a repetitive pattern.

- The final result is a set of frequent itemsets and their periods, represented by a single node labeled \(gF\) at the bottom of the graph.
**PERMINER:** How does it work?

\[
\left( \{gF, dF\}, 2, \{t_1, t_3, t_5, t_8, t_{10}\} \right)
\]

\[
\bigcap t_1, t_3, t_5, t_8, t_{10}
\]

\[
(gF, 2, \{t_1, t_3, t_5, t_8, t_{10}\})
\]

\[
(gF, 3, \{t_5, t_8\})
\]

\[
(gF, 4, \{t_1, t_5\})
\]

\[
(gF, 5, \{t_3, t_5, t_8, t_{10}\})
\]

**Period Computation**

\[
\{gF\}
\]
**PerMiner:** How does it work?

\[
\{gF, dF\}
\]

Recursive call

\[
\bigcap t_1, t_3, t_5, t_8, t_{10}
\]

\[
\{gF, 2, \{t_1, t_3, t_5, t_8, t_{10}\}\}
\]

\[
(gF, 3, \{t_5, t_8\})
\]

\[
(gF, 4, \{t_1, t_5\})
\]

\[
(gF, 5, \{t_3, t_5, t_8, t_{10}\})
\]

**Period Computation**

\[
\{gF\}
\]

\[
gF
\]
## Scalability Experiments

### Experimental set-up
- **PerMiner** implemented in C++
- Run on a multiprocessor computing server:
  - 4 Intel Xeon X7560 processors (8 cores each) 2.27 GHz 64 GB RAM.

### Real Data
- HNDTest Application: Test application for STMicroelectronics middleware for set-top boxes.
- Execution trace of a video playback: 528,360 events, 72 distinct events.
- Split into 10 ms intervals: 15,000 transactions, 35 items/transaction.
Experimental set-up: 1 core, 15,000 transactions, 35 items/transaction.
Scalability Experiments (Parallelization)

- Experimental set-up: 1 to 32 cores, minimum support 10%, 15,000 transactions, 35 items/transaction.

\[
\text{speedup}_n = \frac{\text{sequential execution time}}{\text{execution time with } n \text{ threads}}
\]
Outline

1. Introduction
   - Context and Motivation

2. Formal Framework
   - Frequent Periodic Patterns
   - Core Periodic Concepts
   - PerMiner Algorithm
   - Scalability Experiments

3. Its Application
   - Usage Guidelines
   - Use Cases

4. Conclusions
   - Conclusions
   - Future Work
**Usage Guidelines**

- **Execution Trace**
- **Preprocessing**
- **PERMINER**
- **Postprocessing**

**Special case of KDD methodology**

- Software developers are not familiar with data mining techniques.
- A methodology gives developers the necessary guidelines to exploit this new technique.
Preprocessing

Consists in transforming an execution trace into a transactional database by splitting the trace in a sequence of sets of elements.

- Which information is important?
  - GetFrame, 135_GetFrame, GetFrame_OK, ...

- Which splitting criterion better suits the required analysis?
  - We proposed two methods: Time interval and function name.
  - Domain specific knowledge might propose better suited methods.
How to analyze \textsc{PerMiner} results?

- Manually is not manageable.
- Visualization and analysis tools are needed.
- We have implemented two tools:
  - Analysis tool: Competitors Finder.
  - Visualization tool: \textit{CPCViewer}.
Competitors Finder

**Calculating Competition Ratio**

- Competition Ratio = 100% - (co-execution + co-gap)
- If Competition Ratio \( \geq \) Minimum Competition Ratio \( \Rightarrow \) Competitors!
CPCViewer

Itemsets Hierarchical View

Itemsets Radial View

Periodicity Detailed View

Periodicity Overview

Pattern Visualization

Overview

Detailed View

Periodicit y

Itemsets

Patricia López Cueva

Thesis Defense

8th July, 2013
1st Use Case: HNDTest Application

- **STAPI**: set-top box middleware.
- **STi7200**: set-top box SoC.
- **Tracing**: *KPTrace* kernel module.
- **Trace split into 1 ms intervals.**
- **PerMiner (10%)**: 758 CPCs in 195 s.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPP</td>
<td>18,459</td>
</tr>
<tr>
<td>PC</td>
<td>51,446</td>
</tr>
<tr>
<td>CPC</td>
<td>758</td>
</tr>
</tbody>
</table>
1st Use Case: HNDTest Application

Discovered conflict between the application and the system (USB port)

- **Interrupt_16**: processor clock interrupt.
- **Interrupt_168**: USB interrupt.
- **HNDTest_try_to_wake_up**: system call (**try_to_wake_up**).

**Pattern 1**: Interrupt_16, Interrupt_168

**Pattern 2**: HNDTest_try_to_wake_up
2nd Use Case: GStreamer Application

Multimedia application using GStreamer multimedia framework.

- **Orly STiH416 multi-core MPSoC.**
- **Tracing:** *KPTrace* kernel module.
- **Trace split into 32 ms intervals.**
- **PERMINER (10%):** 787 CPCs in 28 s.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPP</td>
<td>3,086,321</td>
</tr>
<tr>
<td>PC</td>
<td>21,588</td>
</tr>
<tr>
<td>CPC</td>
<td>787</td>
</tr>
</tbody>
</table>

**Figure:** Orly SoC Block Diagram
2nd Use Case: GStreamer Application

Delay in flushing interrupt causing buffer overflow

- Mixer maps audio samples every 32ms.
- Expected undisrupted pattern of period 1.
- Found regular gaps in periodicity.
Outline

1. Introduction
   - Context and Motivation

2. Formal Framework
   - Frequent Periodic Patterns
   - Core Periodic Concepts
   - PerMiner Algorithm
   - Scalability Experiments

3. Its Application
   - Usage Guidelines
   - Use Cases

4. Conclusions
   - Conclusions
   - Future Work
Objective
- Help developers in debugging multimedia embedded applications.
- Proposed a new approach that makes use of periodic pattern mining.

Pattern Mining
- Defined condensed representation of frequent periodic patterns: Core Periodic Concepts (CPC).
- Implemented efficient CPC miner algorithm \texttt{PerMiner}.

Embedded Systems
- Given some guidelines to use our approach.
- Developed postprocessing tools: \texttt{CPCViewer} and Competitors Finder.
Future Work

Pattern Mining
- Define enumeration strategy based on items and periods.
- Explore different types of periodic patterns: sequences, graphs, etc.
- Include domain knowledge in the mining process (SoCTrace, Leon Fopa PhD).

Analysis
- Automatic detection of anomalies.
- Definition of a full methodology (CIFRE, Oleg legorov PhD).
Questions?
Triadic Approach

Periodic Triadic Context \((I, P, T, Y)\)

- \(I\) set of items.
- \(P\) set of periods.
- \(T\) set of transactions.
- \(Y\) ternary relation, \(Y \subseteq I \times P \times T\).

Example (min\_sup = 2)

\[I = \{gF, dF, I16, SI16, \ldots\}\]
\[P = \{1..5(|D|/\text{min\_sup})\}\]
\[T = \{t_1, ..., t_{10}\}\]
\[Y = \{(gF, 2, t_1), (gF, 2, t_3), (gF, 2, t_5), (gF, 2, t_8), (gF, 2, t_{10}), (dF, 2, t_1), (dF, 2, t_3), (dF, 2, t_5), \ldots\}\]
A triple \((I, P, T)\) is **frequent** if \(I \neq \emptyset\), \(P \neq \emptyset\) and \(|T| \geq \text{min}_\text{sup}\).

A frequent triple \((I, P, T)\) is a **periodic concept** if none of its three components can be enlarged without violating the condition \(I \times P \times T \subseteq \mathcal{Y}\).

Example: \(T_1(\{gF, dF\}, \{2\}, \{t_1, t_3, t_5, t_8, t_{10}\})\).

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Periods</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Transactions</td>
<td>t_1</td>
<td>t_2</td>
</tr>
<tr>
<td>gF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>dF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>\ldots</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Periods</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Transactions</td>
<td>t_1</td>
<td>t_2</td>
</tr>
<tr>
<td>gF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>dF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>\ldots</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Connectivity Properties of Core Periodic Concepts

Definition

\(\text{tidlist}(X, p)\) returns the list of all \(t\) transactions such that \((X, p, t) \in \mathcal{Y}\).

Theorem

A periodic concept \((X, P, T)\) is a core periodic concept if and only if for all \(p \in P\) it is true that \(\text{tidlist}(X, p) = T\).

Definition

A triple \((X, P, T)\) is fully connected if \(\forall p \in P\) and \(\forall t \in T\) there exist \(t' \in T \setminus \{t\}\) such that the distance between \(t\) and \(t'\) is equal to \(p\).

Proposition

A core periodic concept is fully connected.
1 procedure PerMiner \((D, min\_sup)\);

Data: dataset \(D\), minimum support threshold \(min\_sup\)

Result: Output all Core Periodic Concepts that occur in \(D\)

2 begin

3 \hspace{1em} \textbf{if} \hspace{0.5em} |D| \geq min\_sup \hspace{0.5em} \textbf{then}

4 \hspace{1em} \bot_{clo} \leftarrow \bigcap_{t \in D} t

5 \hspace{1em} \textbf{output} (\bot_{clo}, \{1..|D|/2\}, D)

6 \hspace{1em} D_{\bot_{clo}} = \{t \setminus \bot_{clo} \mid t \in D\}

7 \hspace{1em} \textbf{foreach} \hspace{0.5em} e \in \mathcal{I} \hspace{0.5em} \textbf{with} \hspace{0.5em} e \notin \bot_{clo} \hspace{0.5em} \textbf{do}

8 \hspace{2em} \textbf{perIter} (\bot_{clo}, D_{\bot_{clo}}, e, \emptyset, min\_sup)
procedure perIter($X, D_X, e, el, min\_sup$);

Data: Itemset of a discovered CPC $X$, reduced dataset $D_X$, item $e$, exclusion list $el$, minimum support threshold $min\_sup$.

Result: Output all Core Periodic Concepts whose itemset is prefixed by $X$ and whose transactions are in $D_X$, with minimal support $min\_sup$.

begin

   $A := \{e\}$
   $B := getPeriods(tidlist(A), min\_sup)$ /* Period computation */
   $B' := B \setminus \{b | \exists b' \in B \text{ such that } b.occ \subset b'.occ\}$
   $G := group(B')$ /* Closure computation */
   $S := \emptyset$
   foreach $g \in G$ do
      $A' := \cap_{t \in g.occ} t$
      $S := S \cup (A', g.periods, g.occ)$
      $S := filter(S)$; /* First parent test */
      $new\_el := el$
      $enum := \emptyset$ /* Itemset enumeration */
      foreach $(A', P, T) \in S$ do
         if $max\_elem(A') = e$ then
            $Q = X \cup A'$
            if $el\_test(Q, el)$ then
               output $(Q, P, T)$
               if $Q \notin enum$ then
                  $D_Q := reduce(D_X, Q, e, min\_sup)$ /* Dataset Reduction */
                  foreach $i \in I$ with $i < e$ and $i \notin Q$ do
                     perIter($Q, D_Q, i, new\_el, min\_sup$)
                     $enum := enum \cup Q$
               $new\_el := new\_el \cup Q$
            endif
         endif
      endforeach
   endforeach

Algorithm 1: Iterative CPC Generator

Patricia López Cueva
Thesis Defense 8th July, 2013 6 / 12
function getPeriods(T, min_supp)
    Data: Transaction list T, minimum support threshold min_supp
    Result: A list of tuples (period, transaction list of the period)
    B ← ∅
    foreach period ∈ [1..|D|/min_supp] do
        b.occς ← ∅
        b.periods := period
        i := 0
        while i < (|T| − 1) do
            if T[i].checked == false then
                j := i + 1
                while j < |T| AND (T[j] − T[i]) <= period do
                    if (T[j] − T[i]) == period then
                        b.occς := b.occς ∪ i; T[i].checked := true
                        b.occς := b.occς ∪ j; T[j].checked := true
                        k := j + 1
                        while k < |T| AND (T[k] − T[j]) <= period do
                            if (T[k] − T[j]) == period then
                                b.occς := b.occς ∪ k; T[k].checked := true
                                j := k
                                k := k + 1
                            j := j + 1
                        i := i + 1
                    else
                        j := j + 1
                end if
            end if
        end while
        if |b.occς| >= min_supp then
            B := B ∪ b −
        end if
    end foreach
    return B
end function
function group($B$)  
\textbf{Data:} List of tuples (period, transaction list of the period) $B$  
\textbf{Result:} A list of tuples grouped by transaction list  
\begin{verbatim}
  foreach $b, b' \in B$ do  
    if $b.occ$ == $b'.occ$ then  
      $b.periods := b.periods \cup b'.periods$
      $B := B \setminus b'$
  
  return $B$
\end{verbatim
function el_test(Q, el)
  Data: Itemset Q, Exclusion list el
  Result: True if none of the elements in el is included in Q. False otherwise.
  foreach X ∈ el do
    if X ⊂ Q then
      return False
  return True
end function
1 function \textit{filter}(S)
   Data: List of CPCs $S$
   Result: A filtered list of CPCs
2 foreach $(A, P, C), (A', P', C') \in S$ do
3     if $A \subset A'$ then
4         \begin{align*}
4         & S := S \setminus (A', P', C') \quad \end{align*}
5     return $S$
6 end function
function reduce(D_{\chi}^{reduced}, A', e, \text{min\_sup})

Data: Database D_{\chi}^{reduced}, Itemset A', element e, minimum support threshold \text{min\_sup}

Result: Reduced Database of A': D_{A'}^{reduced}

1. \(D_{A'}^{reduced} = D_{\chi}^{reduced}[e]\)

2. \(\text{foreach } i \in \mathcal{I} \text{ do} \quad /\!* \text{All items of } \mathcal{I} \text{ with support smaller than } \text{min\_sup} \text{ are removed from the database } */\)

3. \(/\!* \text{min\_sup} \text{ are removed from the database } */\)

4. \(\text{if } \text{support}(i) < \text{min\_sup} \text{ then}\)

5. \(\quad \text{Suppress } i \text{ from all transactions in } D_{A'}^{reduced}\)

6. \(\text{foreach } i \in A' \text{ do} \quad /\!* \text{All items of } A' \text{ are removed from the database } */\)

7. \(\quad \text{Suppress } i \text{ from all transactions in } D_{A'}^{reduced}\)

8. return \(D_{A'}^{reduced}\)

9. end function
Why not a closure operator?

- **Closed Patterns:**
  - Offer a reduced representation of the set of frequent patterns.
  - but **closure operators** based on binary relations.

- **Periodic Patterns:**
  - Ternary relation: the item \( i \) is found in the transaction \( t \) with period \( p \).

- **Closure operator for n-ary relations with \( n > 2 \).**
  - "It is no longer possible to enumerate one attribute domain (usually items) and compute the rest of the pattern thanks to a Galois connection" [CERF09].