

RULEGROWTH: Mining Sequential Rules Common to Several Sequences by Pattern-Growth

Philippe Fournier-Viger¹

Roger Nkambou²

Vincent Shin-Mu Tseng¹

¹National Cheng Kung University (Taiwan)

²University of Québec in Montréal (Canada)



UQAM

March 2011 - **ACM SAC 2011** - Taichung

Introduction

- Many **databases** contain large amount of **temporal information**.
- It is a challenge to develop algorithms for discovering useful temporal patterns in these databases.
- Example:
 - to predict stock market fluctuations,
 - to understand and predict consumer behavior in online web stores,
 - to predict the effect of medicines on patients.
- In this paper, we are interested by **sequence databases** containing sequences of discrete events (e.g. protein sequences and clicks sequence on websites).

Sequence Database

- Each **sequence** is an time-ordered list of itemsets.
- An **itemset** is an unordered set of items (symbols), considered to occur simultaneously.

| ID | Sequences |
|-------------|--|
| <i>seq1</i> | $\{a, b\}, \{c\}, \{f\}, \{g\}, \{e\}$ |
| <i>seq2</i> | $\{a, d\}, \{c\}, \{b\}, \{a, b, e, f\}$ |
| <i>seq3</i> | $\{a\}, \{b\}, \{f\}, \{e\}$ |
| <i>seq4</i> | $\{b\}, \{f, g\}$ |

Sequential Pattern Mining (SPM)

- SPM is probably the most popular set of techniques for discovering temporal patterns in sequence databases.
- SPM finds subsequences that are common to more than *minsup* sequences.
- SPM is limited for making **predictions**. For example, consider the pattern $\{x\},\{y\}$. It is possible that y appears frequently after an x but that there are also many cases where x is not followed by y .
- For **prediction**, we need a measurement of the confidence that if x occurs, y will occur afterward.

Sequential Rule Mining (SRM)

- For prediction, an alternative is **Sequential Rule Mining**.
- A **sequential rule** typically has the form $X \rightarrow Y$ and has a *confidence* and a *support*.
- Several algorithms. e.g. Manila et al. (1997), Hamilton & Karimi (2005), Hsieh (2006), Deogun (2005). But mostly for discovering rules in a single sequence.
- **SRM has several applications:** stock market analysis (Das et al., 1998; Hsieh et al., 2006), weather observation (Hamilton & Karimi, 2005), drought management (Harms et al. 2002), alarm analysis, etc.

Mining Sequential Rules in Sequence Databases

- A **sequential rule** $X \Rightarrow Y$ is a relationship between two disjoint and non empty itemsets X, Y .
- A sequential rule $X \Rightarrow Y$ has **two properties**:
 - **Support**: the number of sequences where X occurs before Y , divided by the number of sequences.
 - **Confidence** the number of sequences where X occurs before Y , divided by the number of sequences where X occurs.
- **The task**: finding all **valid rules**, rules with a support and confidence not less than user-defined thresholds *minSup* and *minConf* (Fournier-Viger, 2010).

An example of Sequential Rule Mining

Consider $minSup= 0.5$ and $minConf= 0.5$:

| ID | Sequences |
|-------------|--|
| <i>seq1</i> | $\{a, b\}, \{c\}, \{f\}, \{g\}, \{e\}$ |
| <i>seq2</i> | $\{a, d\}, \{c\}, \{b\}, \{a, b, e, f\}$ |
| <i>seq3</i> | $\{a\}, \{b\}, \{f\}, \{e\}$ |
| <i>seq4</i> | $\{b\}, \{f, g\}$ |

→

| ID | Rule | Support | Confidence |
|-----|---------------------------------|---------|------------|
| r1 | $\{a, b, c\} \Rightarrow \{e\}$ | 0.5 | 1.0 |
| r2 | $\{a\} \rightarrow \{c, e, f\}$ | 0.5 | 0.66 |
| r3 | $\{a, b\} \rightarrow \{e, f\}$ | 0.5 | 1.0 |
| r4 | $\{b\} \rightarrow \{e, f\}$ | 0.75 | 0.75 |
| r5 | $\{a\} \rightarrow \{e, f\}$ | 0.75 | 1.0 |
| r6 | $\{c\} \rightarrow \{f\}$ | 0.5 | 1.0 |
| r7 | $\{a\} \rightarrow \{b\}$ | 0.5 | 0.66 |
| ... | ... | ... | ... |

A sequence database

Some rules found

Current Algorithms

- **CMRules**: An association rule mining based algorithm for the discovery of sequential rules.
- **CMDeo**: An Apriori based algorithm for the discovery of sequential rules.
- **Limitation**: Both algorithms use a « generate-candidate-and-test » approach that may generate a large amount of candidates for dense datasets. Many candidates do not appear in the database.

RuleGrowth

- Main idea:
 1. First, scan database to find frequent items. e.g. {a, b, c, d ...}
 2. For each pairs of such items, try to create a rule with only two items. e.g. {a}⇒{b}.
 3. Then, find larger rules by recursively scanning the datatabase for adding a single item at a time to the left or right part of each rule (*left* and *right expansions*).
e.g. {a,c} ⇒{b} , then {a,c,d} ⇒{b}, etc.
 4. Each rule created is tested to see if it is **valid**.

RuleGrowth (2)

- When a rule should be expanded?

Property: Adding an item to a rule results in a rule having a support that is lower or equal.

Therefore, a rule should be expanded only if has the minimum support.

- When a rule should be outputted?

If the support and confidence and support are higher or equal to *minsup* and *minconf*.

RuleGrowth (3)

- How to choose items for performing left expansions of a rule $X \Rightarrow Y$?
 - Scan the sequences containing the rule and note items appearing in at least *minsup* sequences before the last occurrence of Y.
- How to choose items for performing right expansions of a rule $X \Rightarrow Y$?
 - Scan the sequences containing the rule and note items appearing in at least *minsup* sequences after the first occurrence of X.

RuleGrowth (4)

- How to avoid generating the same rules twice?
 - Add only an item to the left/right part of a rule if the item is larger than all items already in the left/right part.
 - Do not allow performing a left expansion after a right expansion. But allow performing a right expansion after a left expansion

RuleGrowth (5)

Optimization: During the first database scan, record the first and last occurrence of each item for each sequence.

- This allows to create initial rules very efficiently.
- This allows to avoid scanning sequences completely when searching for items for expansions.

Optimization 2: The set of sequences containing X , Y , and $X \Rightarrow Y$ is maintained for each rule $X \Rightarrow Y$ so that the confidence can be calculated efficiently.

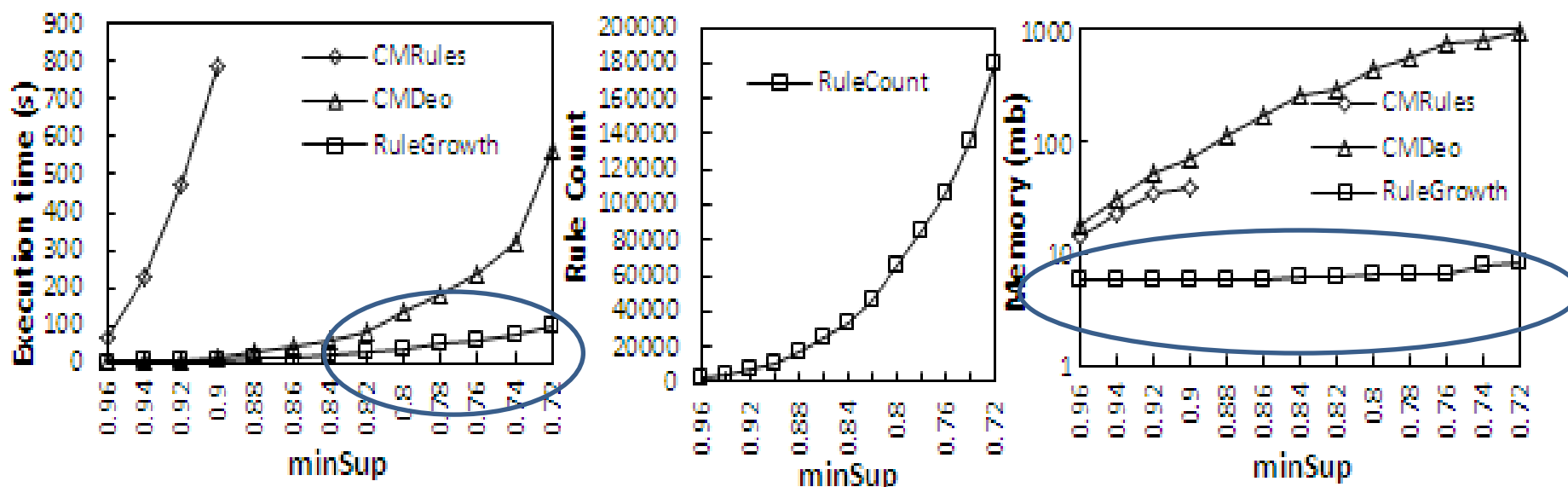
Performance Evaluation

- RuleGrowth, CMRules and CMDEO.
- Java, 1GB of RAM
- Three real-life public datasets.

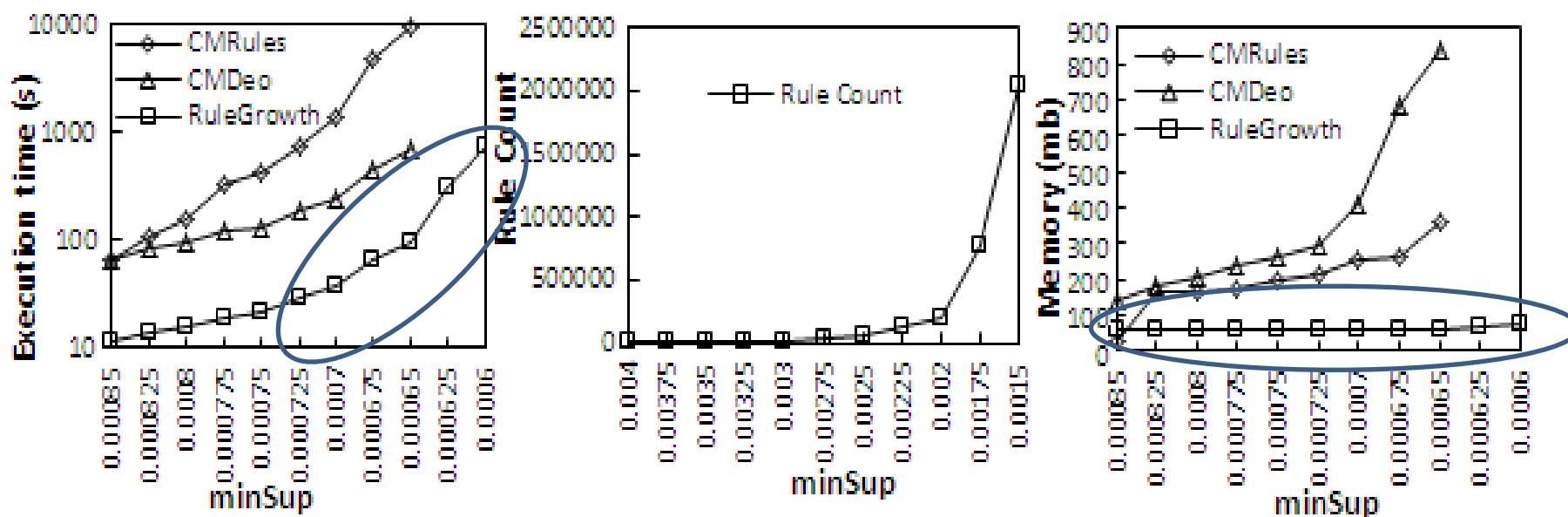
| | Kosarak | BMS | Toxin-Snake |
|--|----------------|------------|--------------------|
| Sequence count | 70,000 | 59,601 | 163 |
| Item count | 21,144 | 497 | 20 |
| Average item count by sequence | 7.97 | 2.51 | 60.61 |
| Average different item count by sequence | 7.97 | 2.51 | 17.84 |

Influence of *minsup*

Snake

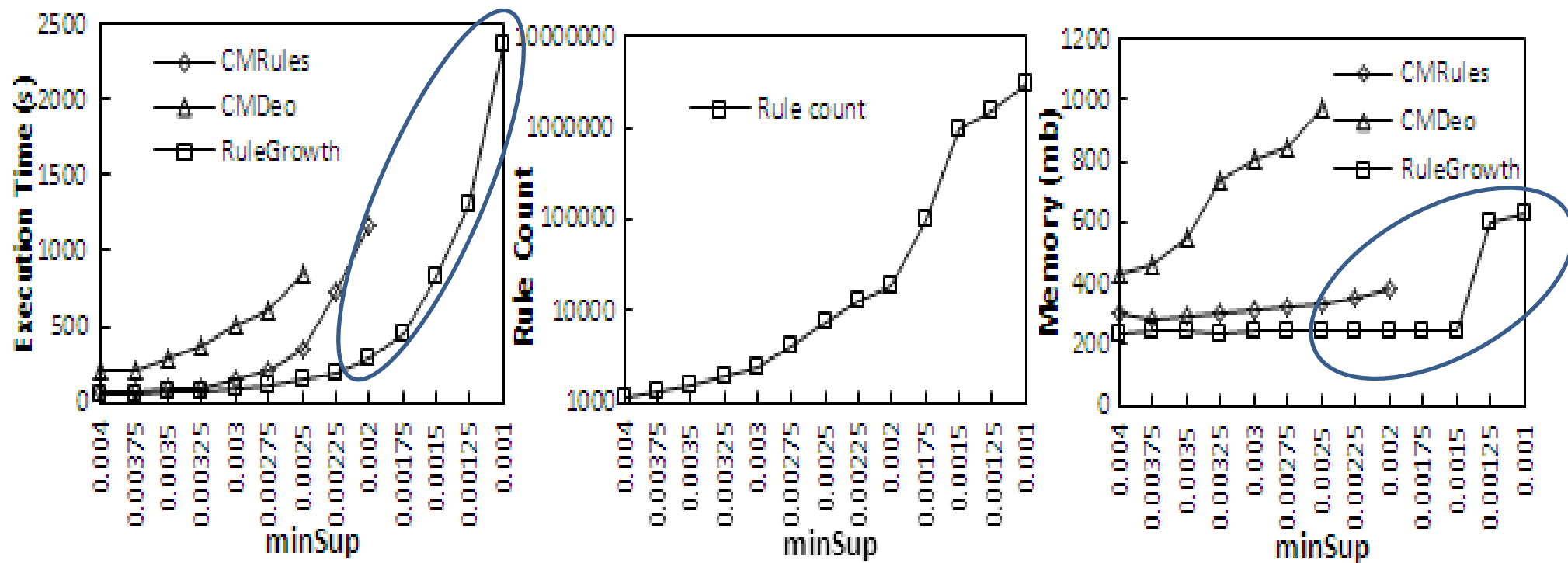


BMS

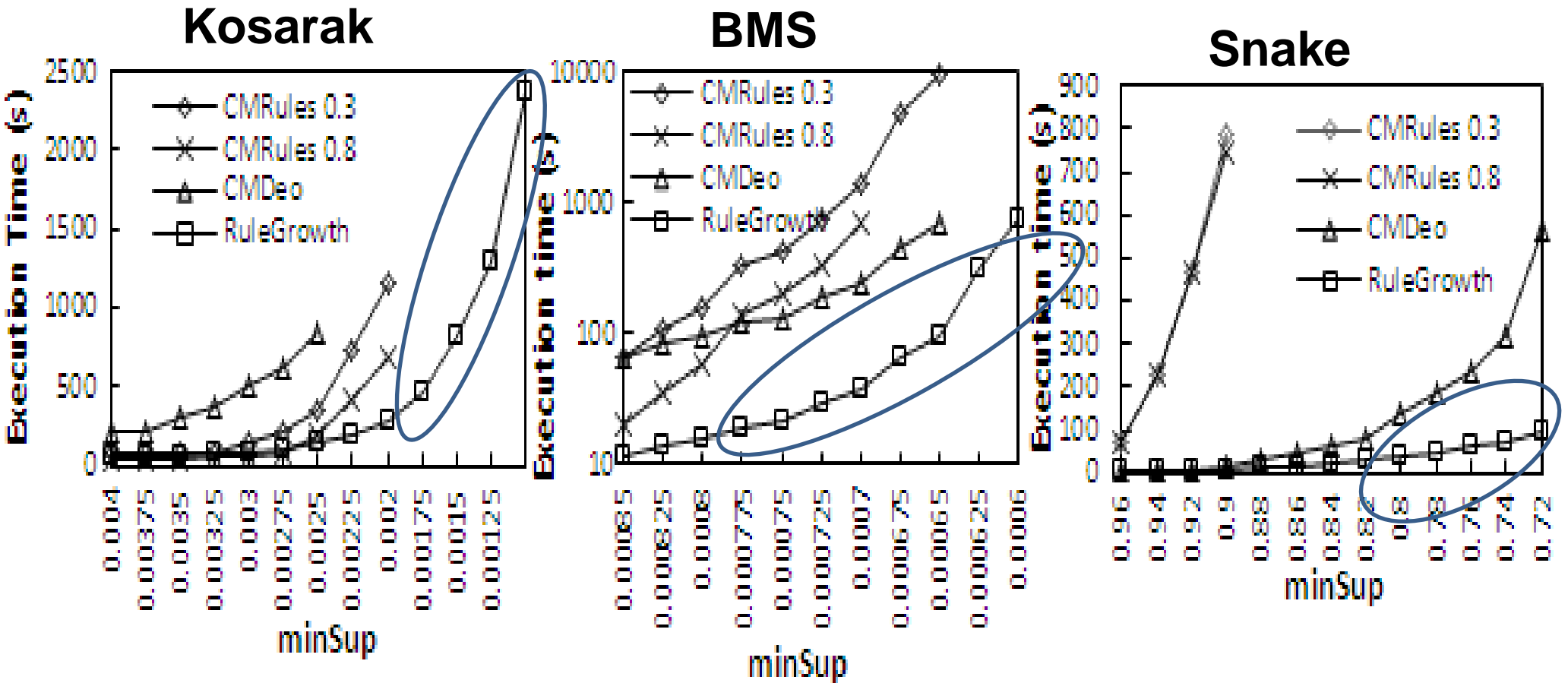


Influence of *minsup*

Kosarak



Influence of *minconf*



Conclusion

RuleGrowth,

- Is a novel algorithm for mining sequential rules common to several sequences,
- outperforms **CMRules** and **CMDeo** in terms of execution time and memory usage.

The **Java source code** can be downloaded as part of the **Sequential Pattern Mining Framework** at

<http://goo.gl/xat4k>

Thank you. Questions?



Thanks to the organizers of the SAC 2011 Conference & DM track!

Thanks to the NSERC and FQRNT funding programs.

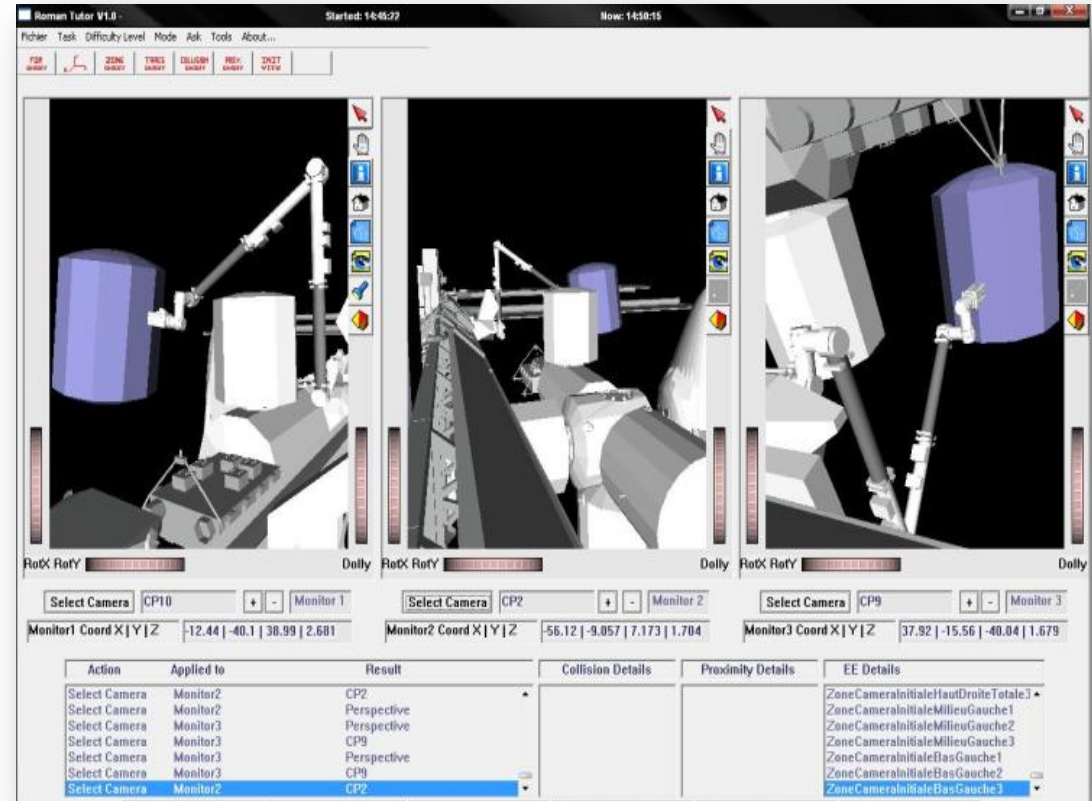


Fonds de recherche
sur la nature
et les technologies



Application in a Cognitive Agent

- The **CTS cognitive agent** has predefined behaviors.
- Each execution of CTS is recorded as a sequence in a **sequence database**
- **Items** are actions and perceptions of CTS
- **Rules** are used for making predictions about which behavior will be successful with learners, and adapt the behavior of CTS accordingly.



CanadarmTutor

