Constraint modeling and solving for Data Mining

Tias Guns

Declarative Languages & AI lab,
KU Leuven, Belgium

With thanks to Luc De Raedt, Siegfried Nijssen and others

MoKMaSD 2015
Constraint solving

“Combinatorial problem = Model + Solve”

**Model** = specification of constraints over variables

**Solve** = search for satisfying/optimal solutions
  - generic handling of variables and constraints
  - efficient propagation of individual constraints
Data Mining

“Extracting useful information from data”

- Machine learning: extracting predictive models
- Clustering: extracting meaningful groups
- Pattern mining: extracting regularities
- Recommender systems: extracting preferences
Solving data mining problems

Specific methods/algorithms for specific problems

Limited flexibility:

- New problems rarely fit existing methods well
- Tedious programming & hacks
- Refining solution methods is hard, but typical in the knowledge discovery cycle
Constraint solving can help!

- Complex constraints
- Reusing solving technology
- Adding/removing (user) constraints
- Exhaustive, optimal

Used in a wide range of problems in data mining
Active research directions

- **Pattern Mining**  
P. Boizumault, B. Cremilleux, L. De Raedt, T. Guns, S. Jabbour, M. Jarvisalo, S. Loudni, S. Nijssen, B. O'Sullivan, A. Kemmar, ...

- **Clustering**  
B. Babaki, I. Davidson, T.B.H. Dao, K.C. Duong, O. du Merle, S. Gilpin, P. Hansen, S. Nijssen, C. Vrain, V. Grossi, A. Monreale...

- **Structure learning**  
C. Bessiere, J. Cussens, O. Grinchtsein, M. Heule, T. Jaakkola, M. Meila, B. O'Sullivan, D. Sontag, P. Van Beeck, S. Verwer, ...
Pattern Mining

“Finding regularities in data”

Text Mining

Well, there’s egg and spam; egg sausage and spam; spam and bacon; egg bacon and spam; egg bacon sausage and spam; spam bacon sausage and spam; spam egg spam spam bacon and spam; spam sausage spam bacon spam tomato and spam; spam spam spam egg and spam; spam spam spam spam baked beans spam spam spam spam; or Lobster Thermidor, a Crevette with a mornay sauce served in a Provençale manner with shallots and aubergines garnished with truffle pate, brandy and spam.

Graph Mining

Itemset Mining
Constraint-based Itemset Mining

- Fundamental enumeration problem
- Well studied
- Many constraints
- Many applications
Frequent Itemset Mining

Find: set of *items* appearing frequently (enumeration)

\[
\text{cover}(\{\text{item1}, \text{item2}\}) = \{\text{set1}, \text{set2}\}
\]

\[
\text{frequency}(\{\text{item1}, \text{item2}\}) = |\{\text{set1}, \text{set2}\}| = 2
\]
CP for Itemset Mining

<table>
<thead>
<tr>
<th></th>
<th>i1</th>
<th>i2</th>
<th>i3</th>
<th>i4</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>t3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Coverage: \( \forall T_t: T_t = 1 \Leftrightarrow \text{set}(I_1, \ldots, I_n) \subseteq \text{set}(\text{row}_t) \)

Frequency: \( \sum_t T_t \geq \text{Freq} \)
CP for Itemset Mining

coverage: \[ \forall T_t: \quad T_t = 1 \iff \sum_i I_i (1 - D_{ti}) = 0 \]

frequency: \[ \forall I_i: \quad I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq Freq \]

[L. De Raedt, T. Guns, S. Nijssen, KDD 2008]
% params/data
int: MinFreq;
int: NrI; int: NrT;
array [1..NrT,1..NrI] of 0..1: TDB;

% vars
array[1..NrI] of var 0..1: Items;
array[1..NrT] of var 0..1: Trans;

% Trans covered if Itemset subset of the transaction
constraint forall(t in 1..NrT) (    
    Trans[t] == 1 <-> sum(i in 1..NrI) (Items[i]*(1-TDB[t,i])) = 0 );

% each Item must be covered by sufficiently many Trans
constraint forall(i in 1..NrI) (    
    Items[i] == 1  -> sum(t in 1..NrT) (Trans[t]*TDB[t,i]) >= MinFreq );

[L. De Raedt, T. Guns, S. Nijssen, KDD 2008]
More constraints

• Coverage (required)

• Frequent

• Maximal

• Closed

• Delta-closed

• ...

\[ T_t = 1 \iff \sum_i I_i (1 - D_{ti}) = 0 \]

\[ I_i = 1 \implies \sum_t T_t D_{ti} \geq \text{Freq} \]

\[ I_i = 1 \iff \sum_t T_t D_{ti} \geq \text{Freq} \]

\[ I_i = 1 \iff \sum_t T_t (1 - D_{ti}) = 0 \]

\[ I_i = 1 \iff \sum_t T_t (1 - \delta - D_{ti}) = 0 \]

+ combinations!
## Generality

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum frequency</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Maximum frequency</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Emerging patterns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

| Condensed Representations                |          |           |             |               |               |
| Maximal                                  | X        | X         | X            | X             |
| Closed                                   | X        | X         |             |               |
| $\delta$ – Closed                        |          |           | X            |               |

| Constraints on syntax                    |          |           |             |               |               |
| Max/Min total cost                       |          |           | X            | X             |
| Minimum average cost                     |          |           | X            |               |
| Max/Min size                             | X        | X         | X            | X             |

| Constraints on labelled data             |          |           |             |               |               |
| Minimum correlation                      |          |           | X            |               |
| Maximum correlation                      |          |           | X            |               |
Basic Frequent Itemset Mining

Minimum support

Specialised systems

CP (Gecode)
Constraint-based Itemset Mining

![Graph showing runtime vs max average cost for different systems. The graph includes lines for FIM_CP_1%, FIM_CP_5%, FIM_CP_10%, PATTER_1%, PATTER_5%, PATTER_10%, and LCM_10%. The x-axis represents max average cost, and the y-axis represents runtime in seconds. The graph highlights 'Specialised systems' and notes CP (Gecode).]
Take away message 1.

Constraint Programming for Itemset Mining:

- Intuitive, reasonably compact encoding
- Generic: many constraints can be expressed
- Effective in case of tight constraints

Many extensions (not in this talk):

- Pattern set mining
  [M. Khiari, P. Boizumault, B. Cremilleux, ISMIS 2011]
  [T. Guns, S. Nijssen, L. De Raedt, TKDE 2013]

- Skypatterns / multi-objective
  [A. Soulet, C. Raïssi, M. Plantevit, B. Crémilleux, ICDM 2011]
  [W. Ugarte, P. Boizumault, S. Loudni, B. Cremilleux, ECAI 2014]

- SAT, BDD, ASP solvers
  [JP. Metivier, P. Boizumault, B. Cremilleux, M. Khiari, S. Loudni, IDA 2012]
  [H. Cambazard, T. Hadzi, B. O'Sullivan, ECAI 2010]
  [M. Jarvisalo, LPNMR 2011]
So far only enumeration problems.

What about optimisation problems?
Optimisation: discriminative itemsets

Known as: correlated itemset mining, discriminative itemset mining, contrast set mining, emerging itemsets, subgroup discovery, ...

- **Given**: labelled transactions

- **Find**: the itemset that *best correlates* with the class label

  - : {+, \(\times\)}
  -  : {+, +}
  -  : {+, +}
Correlation function

\[ f(\sum_{t \in P} T_t, \sum_{t \in N} T_t) \geq \Theta \]

Example functions: chi^2, information gain, accuracy, ...

Common property: convex and zero-on-the-diagonal

- Existing pruning technique:
  only uses upper-bound of \( \sum T \)

- Our CP-based propagator:
  uses upper- and lower-bound of \( \sum T \)
  and look-ahead formulation
  \[ I_i=1 \Rightarrow \cdots \]

much stronger propagation!

[T. Guns, S. Nijssen, L. De Raedt, AIJ 2012]
## Correlated itemset mining

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CP</th>
<th>(Cheng et al. 2008)</th>
<th>(Morishita and Sese 2000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>anneal</td>
<td>0.22</td>
<td>22.46</td>
<td>24.09</td>
</tr>
<tr>
<td>australian-credit</td>
<td>0.30</td>
<td>3.40</td>
<td>0.30</td>
</tr>
<tr>
<td>breast-wisconsin</td>
<td>0.28</td>
<td>96.75</td>
<td>0.28</td>
</tr>
<tr>
<td>diabetes</td>
<td>2.45</td>
<td>—</td>
<td>128.04</td>
</tr>
<tr>
<td>heart-cleveland</td>
<td>0.19</td>
<td>9.49</td>
<td>2.15</td>
</tr>
<tr>
<td>hypothyroid</td>
<td>0.71</td>
<td>—</td>
<td>10.91</td>
</tr>
<tr>
<td>ionosphere</td>
<td>1.44</td>
<td>—</td>
<td>&gt;</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>0.92</td>
<td>125.60</td>
<td>46.20</td>
</tr>
<tr>
<td>letter</td>
<td>52.66</td>
<td>—</td>
<td>&gt;</td>
</tr>
<tr>
<td>mushroom</td>
<td>14.11</td>
<td>0.09</td>
<td>13.48</td>
</tr>
<tr>
<td>primary-tumor</td>
<td>0.03</td>
<td>0.26</td>
<td>0.13</td>
</tr>
<tr>
<td>segment</td>
<td>1.45</td>
<td>—</td>
<td>&gt;</td>
</tr>
<tr>
<td>soybean</td>
<td>0.05</td>
<td>0.05</td>
<td>0.07</td>
</tr>
<tr>
<td>splice-1</td>
<td>30.41</td>
<td>1.86</td>
<td>31.11</td>
</tr>
<tr>
<td>vehicle</td>
<td>0.85</td>
<td>—</td>
<td>&gt;</td>
</tr>
<tr>
<td>yeast</td>
<td>5.67</td>
<td>—</td>
<td>781.63</td>
</tr>
</tbody>
</table>

Runtime in seconds

- : out of memory
  > : time-out

[T. Guns, S. Nijssen, L. De Raedt, AIJ 2012]
Take away message 2.

Correlated itemset mining:
- (constrained) optimisation problem
- Novel propagator for correlation functions
- CP encoding+propagator beats state-of-the-art

Open issues:
- Identify other complex functions with good properties?
- Statistical evaluation of patterns common; statistical constraints in CP?
- Strong links to rule learning; potential for interactive classifier construction?

[R. Rossi, S. Prestwich, S. Armagan Tarim, ECAI 2014]
All pattern mining problems:

- Pattern $P = P_1 \; P_2 \; P_3 \; P_4 \; ...$

- Trans:  
  
<table>
<thead>
<tr>
<th>Data:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
</tr>
<tr>
<td>$T_2$</td>
</tr>
<tr>
<td>$T_3$</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Constraints:

- **coverage**: $\forall \; i: T_i \leftrightarrow included(P,X_i)$

- **frequency**: $\sum_i(T_i) \geq V$
Coverage for ...

- **Itemsets** \( P=\{1,2,4\} \quad X_i=\{1,2,3,4,8,9\} \)
  - \( T_t = 1 \iff P \subseteq X_i \)
  - 'subset' constraint in CP (or Boolean encoding)

- **Sequences with wildcards** \( P=\langle 3,*,2 \rangle \quad X_i=\langle 1,3,5,2,2 \rangle \)
  - \( T_t = 1 \iff \exists o : \forall j \, P[j] = X_i[j+o] \)
  - in CP: exists \( o \iff o=1 \) or \( o=2 \) or ...

- **Sequences (standard)** \( P=\langle 3,2 \rangle \quad X_i=\langle 1,3,5,2,2 \rangle \)
  - \( T_t = 1 \iff \exists (e_1 \ldots e_n) : e_1 < \ldots < e_n \land \forall j \, P[j] = X_i[e_j] \)
  - must check all possible \( (e_1 \ldots e_n) \)
Global exist-embedding

∀ \( T_t: T_t = 1 \Leftrightarrow \text{exist-embedding}(S, X_t) \)

\[ \sum_t T_t \geq Freq \]

Global constraint with filtering algorithm:

- \( T_t = 1 \Leftrightarrow \exists(e_1..e_n): e_1 < ... < e_n \land \forall j \quad S[j] = X_i[e_j] \)
- incremental: keep one pointer to last assigned match \( e_j \)
global exists-embedding

S: <E, V, ?, ?, ?>
T1: <X, E, R, V, V, R, R>

T2: <U, T, E, D, D, V, V>

Many symbols can become infrequent after *prefix-projection* → count symbols in projected database, drop infrequent symbols

Same idea as used in famous PrefixSpan algorithm
Other constraints

- Constraints on sequence:
  size, regular expr., …

- Constraints on cover set:
  min_freq, max_freq, discriminative, …

- Constraints on inclusion relation:
  max_gap, min_gap, max_span

- Preferences over the solution set
  closed, maximal, multi-objective (Dominance Programming)
Efficiency
Constraints

Unix user (num. patterns)

JMLR (num. patterns)

iPRG (num. patterns)

FIFA (num. patterns)

Unix user (run time)

JMLR (run time)

iPRG (run time)

FIFA (run time)
Take away message 3.

**Sequence mining**: more complex *coverage* relation

global constraint:

- hides complexity of *coverage* relation
- fast (incremental, PrefixSpan-like)
- good way to hybridize with data mining techniques
  => necessary to be **efficient**

**Flexible**: one new global constraint, other constraints compatible

**Exception**: constraints on *coverage* relation
Active research directions

- Pattern Mining

- Clustering: constraint-based clustering

- *Structure learning*
Clustering: grouping by similarity
Constraint-based clustering

Background knowledge about clusters:

- Type of clustering
  - hierarchical
  - partition-based
  - spectral clustering

- Distance function (distance between two points)

- User constraints
  - Must-link
  - Cannot-link
  - Minimum size
  - ...

Hierarchical clustering

Surveyed as most popular method in practice

General algorithm:

- Initially every instance is separate cluster
- Repeat
  - Compute similarities between clusters
  - Merge two most similar clusters
Hierarchical clustering

Constraints:
- Must-link and cannot-link
- Conditional ML/CL
- Precedence
- Constraints on levels, ...

SAT-based modification (Horn-SAT!):
- Initially every instance is separate cluster
- Repeat
  - Compute similarities between clusters
  - Do until merge found:
    - Query SAT solver: can merge the two most similar clusters?

[S. Gilpin, I. Davidson, KDD 2011]
Hierarchical clustering

Still a local, greedy algorithm...
Can the global optimum be found?

Using Integer Linear Programming:
- Variables $M_{ij}$: what level points $i$ & $j$ are merged at
- Constraints: reflexivity, symmetry, transitivity

\[
\arg \max_{\mathcal{M}, O, Z} \sum_{a, b, c \in \text{Instances}} w_{abc} \times O_{abc}
\]
subject to:

$O, Z, \mathcal{M}$ are integers.
$0 \leq O \leq 1, 0 \leq Z \leq 1$
$1 \leq \mathcal{M} \leq L$
$-L \leq \mathcal{M}(a, c) - \mathcal{M}(a, b) - (L + 1)O_{abc} \leq 0$
$-L \leq \mathcal{M}(a, b) - \mathcal{M}(a, c) - (L + 1)Z_{ab \geq ac} + 1 \leq 0$
$-L \leq \mathcal{M}(b, c) - \mathcal{M}(a, c) - (L + 1)Z_{bc \geq ac} + 1 \leq 0$
$Z_{ab \geq ac} + Z_{bc \geq ac} \geq 1$

- Objective: ultrametric
favoring early joins of similar instances

[S. Gilpin, S. Nijssen, I. Davidson, AAAI 2013]
Hierarchical clustering

[S. Gilpin, S. Nijssen, I. Davidson, AAAI 2013]

ILP formulation: finds optimal solution

- Good for high-value data
- More robust to noise than greedy:

Novel settings:

- relaxing transitivity constraints = hierarchy with overlapping clusters
Take away message 4.

Constraint-based clustering:
- constraint solving already supports complex constraints

Optimal clustering:
- value of computation time vs solution quality
- robustness against local minima/noise

Other clustering settings:
- “Cluster Analysis and Mathematical Programming”  
  [M.R. Rao, JASA 1971]  
  [P. Hansen & B. Jaumard, MP 1997]
- Solving: constrained-based K-means-like,  
  [O. du Merle, P. Hansen, B. Jaumard, N. Mladenovic, JSC 1999]  
  [B. Babaki, T. Guns, S. Nijssen, CPAIOR 2014]
  spectral clustering, ...
- Modeling? Only methods specialised to one objective function?  
  [T.B.H. Dao, K.C. Duong, C. Vrain, ECMLPKDD 2013]
Constraint Solving = Model + Solve

Encoding problem in effective constraints

- Mining algorithms
- Generic CP solvers
- CP + global constraints

Better modeling support for data mining?
A modeling language for pattern mining?

Goals:
- High-level, natural notation (for data miners)
- User-defined constraints
- Efficient solving (with\&without extra constraints)
- Support existing mining algorithms?

Observation1: can formulate as CSP, reuse high-level CSP language?

Observation2: in enumeration problems sometimes better to post-process solution set; do this automatically?
A modeling language for pattern mining?

MiningZinc

- Based on the established MiniZinc language
  - High-level mathematical-like notation
  - User-defined constraints and functions
  - Solver independent (10+ CP solvers & SAT & MIP)

- **Modeling:** Pattern mining specific constrains and functions

- **Solving:** automatic model reformulations;
  Can detect and use specialised mining algorithms (LCM, Eclat, Fpgrowth)
  + post-processing with generic solvers

Example: freq. itemset mining with cost

library with itemset mining specific functions and predicates

```
include "lib_itemsetmining.mzn"

int: Nrl; int: NrT; int: MinFreq;
array[1..NrT] of set of int: TDB;

var set of 1..Nrl: Items;

constraint card(cover(Items, TDB)) >= MinFreq;

array [1..Nrl] of int: Cost;
int: MinCost;

constraint sum(i in Items) (Cost[i]) >= MinCost

solve satisfy;
```
1) Normalize to FlatZinc *(do not flatten lib_itemsetmining.mzn yet)*

2) Apply rewrite rules to:
   - add redundant constraints
   - detect (partial) applicability of specialised algorithms
   - tailor to constraint solvers

3) Collect all feasible rewrite combinations = execution plans

4) Heuristically rank + execute a plan
Rewrite system / decompositions

- Rules for redundant constraints:
  ```
  if set-of-constraints-1 matches with substitution θ,
  then add set-of-constraints-2/θ
  ```

- Rules for specialised algorithms:
  ```
  if set-of-constraints matches with substitution θ,
  then remove set-of-constraints/θ add algo-predicate/θ to exec. plan
  ```

- Rules for CP systems:
  ```
  do optionally transform sets to Booleans
  do re-flatten to solver-specific FZN
  if all constraints supported
  then remove constraints and
    add solver-predicate(all-constraints) to execution plan
  ```
Generating all plans

Depth-first search over all possible rewrite rules, with:

- redundant rules (add constraints) before others (remove constraints)
- never apply the same rule twice
- stop when empty constraint set or when non-empty and no more rules applicable

Ranking plans:

- Divide in 'specialised only', 'hybrid' and 'CP only'
- Order based on global ordering of solvers and nr of constraints
Example

A) Specialised algorithms only
   - Eclat-maxfreq(TDB, 20, 50)
   - LCMv2(TDB, 20) + maxcover(Items, TDB, 40)

B) Hybrid decomposition
   - LCMv2(TDB, 20) + gecode(card(cover(Items, TDB)) =< 40)
   - LCMv2(TDB, 20) + frequency(Items, TDB, S) + gecode(S =< 40)

C) Generic solvers only
   - gecode(...)
   - gecode-bool(...)
   - gecode-bool(...) + redundant
   - or-tools-bool(...)

```
var set of 1..Nrl: Items; array[int] of set of int: TDB;
constraint card(cover(Items, TDB)) >= 20;
constraint card(cover(Items, TDB)) =< 40;
solve satisfy;
```
Comparing CP solvers

frequent itemset mining, no extra constraints

<table>
<thead>
<tr>
<th></th>
<th>Gecode</th>
<th></th>
<th></th>
<th>CPX</th>
<th></th>
<th>or-tools</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>set</td>
<td>set/red</td>
<td>bool</td>
<td></td>
<td>set</td>
<td>bool/red</td>
</tr>
<tr>
<td>soybean</td>
<td>2.65</td>
<td>4.02</td>
<td>2.04</td>
<td>0.58</td>
<td>3.20</td>
<td>3.08</td>
</tr>
<tr>
<td>primary-tumor</td>
<td>2.14</td>
<td>11.61</td>
<td>1.30</td>
<td>2.65</td>
<td>33.20</td>
<td>9.66</td>
</tr>
<tr>
<td>vote</td>
<td>3.32</td>
<td>13.48</td>
<td>2.34</td>
<td>4.16</td>
<td>39.40</td>
<td>20.33</td>
</tr>
<tr>
<td>lymph</td>
<td>92.85</td>
<td>220.76</td>
<td>216.53</td>
<td>211.36</td>
<td>110.85</td>
<td>241.37</td>
</tr>
<tr>
<td>mushroom</td>
<td>580.67</td>
<td>308.30</td>
<td>604.41</td>
<td>81.52</td>
<td>46.42</td>
<td>239.37</td>
</tr>
<tr>
<td>hepatitis</td>
<td>386.88</td>
<td>446.16</td>
<td>385.57</td>
<td>376.07</td>
<td>415.24</td>
<td>573.33</td>
</tr>
<tr>
<td>german-credit</td>
<td>523.58</td>
<td>582.50</td>
<td>446.38</td>
<td>310.30</td>
<td>384.35</td>
<td>587.66</td>
</tr>
<tr>
<td>heart-cleveland</td>
<td>568.66</td>
<td>595.44</td>
<td>482.94</td>
<td>425.74</td>
<td>544.58</td>
<td>693.37</td>
</tr>
<tr>
<td>australian-credit</td>
<td>755.83</td>
<td>757.56</td>
<td>674.09</td>
<td>580.25</td>
<td>565.78</td>
<td>761.22</td>
</tr>
</tbody>
</table>

automatic from high-level specification to low-level effective solving

as before: slower than specialized methods on this task though
Experiments, hybrid solving

frequent itemset mining, with minimum size and closure constraint
Take away message 4.

Modeling:
- Can build on existing high-level languages
- Solver independence:
  - Automatic model rewriting
  - Automatic chaining of CP/DM algorithms: hybridisation

Open questions
- Multiple execution strategies: algorithm selection?
- Problems not fitting standard CP
  - Skyline patterns / multi-objective
  - Dominance / preference over solutions
- Beyond enumeration
Overview

Modeling paradigm already prevalent in A.I.:
SAT, ILP/MIP, Graphical Models, STRIPS, ...

Increased use in data mining too:
• On wide range of tasks
• For handling complex constraints (declarative)
• Reuse of solving technology: can beat state-of-the-art

Creates shift from 'programming' to 'modelling'
→ efficient encoding of problem is crucial
Challenges

Big data vs Smart data vs Valuable data

Nevertheless, scalability:

- Novel encodings or propagators
- Solver technology/search strategy
- Hybridization
- Generic heuristic search?

→ new benchmarks for solvers...

Other tasks: plenty more in pattern mining, clustering, struct. learning

Beyond CSPs? multi-objective, soft constraints, preferences, statistics
Thank you for listening

Questions?

http://dtai.cs.kuleuven.be/CP4IM