A Divide and Revise Method for Geographical Data

Éric Würbel
wurbel@lim.univ-mrs.fr

April 10 2001
1. **Introduction and background**

- An application handling environmental data.
1. Introduction and background

- An application handling environmental data.
- The application involves a revision problem.
1. Introduction and background

- An application handling environmental data.
- The application involves a revision problem.
- A huge amount of data.
1. Introduction and background

- An application handling environmental data.
- The application involves a revision problem.
- A huge amount of data.
- An algorithm which implements the revision operation.
1. Introduction and background

- An application handling environmental data.
- The application involves a revision problem.
- A huge amount of data.
- An algorithm which implements the revision operation.
- Limitations of the algorithm.
1. Introduction and background

• An application handling environmental data.
• The application involves a revision problem.
• A huge amount of data.
• An algorithm which implements the revision operation.
• Limitations of the algorithm.
• A “divise and revise” method.
2. **Application**

2.1. **Goals**

- Application: assessing water heights in a flooded valley.
2. Application

2.1. Goals

- Application: assessing water heights in a flooded valley.
- Research: developing efficient revision algorithms.
2. **Application**

2.1. **Goals**

- Application: assessing water heights in a flooded valley.
- Research: developing efficient revision algorithms.

2.2. **Characteristics of the sources of information**

- Easily accessible and “cheap”.
2. Application

2.1. Goals

- Application: assessing water heights in a flooded valley.
- Research: developing efficient revision algorithms.

2.2. Characteristics of the sources of information

- Easily accessible and “cheap”.
- Uncertain and incomplete.
2.3. The flooded valley
2.4. Sources of information

compartments: constant water height.
2.4. Sources of information

*compartments*: constant water height.

$S_1$ max or min submersion heights.
2.4. Sources of information

\[ S_1 \text{ max or min submersion heights.} \]

- not reliable and incomplete
2.4. Sources of information

compartments: constant water height.

$S_1$ max or min submersion heights.

- not reliable and incomplete

$S_2$ hydraulic relations
2.4. Sources of information

*compartments*: constant water height.

$S_1$ max or min submersion heights.

- not reliable and incomplete

$S_2$ hydraulic relations

- more reliable than $S_1$
3. Knowledge representation

3.1. Propositional encoding

\[ A \]
3. Knowledge representation

3.1. Propositional encoding

\[ A^+ \quad A \quad A^- \]
3. Knowledge representation

3.1. Propositional encoding

\[ A^+ \text{ with domain } D_{A^+} = \{a_1, \ldots, a_n\} \]
3. Knowledge representation

3.1. Propositional encoding

A+ with domain $D_{A^+} = \{a_1, \ldots, a_n\}$

- $A_{a_1}^+ \vee \ldots \vee A_{a_n}^+$: enumeration of the domain.
- $\{\neg A_j^+ \vee \neg A_k^+, j, k \in \{a_1, \ldots, a_n\}, j \neq k\}$: mutual exclusion
3.2. $S_2$ source
3.2. $S_2$ source

\[ A^+ \geq B^+ \]
\[ A^+ > B^- \]
\[ A^- \geq B^- \]
3.2. \( S_2 \) source

\[
\begin{align*}
A^+ &\geq B^+ \\
A^+ &> B^- \\
A^- &\geq B^-
\end{align*}
\]

\( A^+ > B^- \) : defines a relation \( R \subseteq D_{A^+} \times D_{B^-} \).
3.2. $S_2$ source

\[
\begin{align*}
A^+ & \geq B^+ \\
A^+ & > B^- \\
A^- & \geq B^-
\end{align*}
\]

$A^+ > B^-$: defines a relation $R \subseteq D_{A^+} \times D_{B^-}$.

We code tuples from $(D_{A^+} \times D_{B^-}) \setminus R$:

- $\forall (x, y) \in (D_{A^+} \times D_{B^-}) \setminus R$, we produce a clause $\neg A^+_x \lor \neg B^-_y$. 
3.3. $S_1$ source

\[ A^+ = a_i \]

\[ A^- = a_j \]

\[ B^+ = b_k \]
3.3. \( S_1 \) source

\[
\begin{align*}
A^+ &= a_i \\
A^- &= a_j \\
B^+ &= b_k
\end{align*}
\]

\[
\begin{align*}
A^+_{a_i} & \\
A^-_{a_j} & \\
B^+_{b_k}
\end{align*}
\]
4. Revision problem

4.1. Definition of the problem

- $S_1$ is consistent, $S_2$ is consistent, but

$$S_1 \cup S_2 \text{ is inconsistent}$$
4. Revision problem

4.1. Definition of the problem

• $S_1$ is consistent, $S_2$ is consistent, but

$$S_1 \cup S_2 \text{ is inconsistent}$$

• $S_2$ is preferred to $S_1$
4. Revision problem

4.1. Definition of the problem

• \( S_1 \) is consistent, \( S_2 \) is consistent, but \( S_1 \cup S_2 \) is inconsistent

• \( S_2 \) is preferred to \( S_1 \)

• New epistemic state \( S_1 \star S_2 \)
4.2. Consistent case

\[ S_1 \cup S_2 \text{ consistent} \]
4.2. Consistent case

\[ S_1 \cup S_2 \text{ consistent} \]

\[ S_1 \ast S_2 = S_1 \cup S_2 \]
4.3. Inconsistent case

\[ S_1 \cup S_2 \text{ inconsistent} \]
4.3. Inconsistent case

\[ S_1 \cup S_2 \text{ inconsistent} \]

\[ S_1 \star S_2 \]
4.3. Inconsistent case

\[ S_1 \cup S_2 \text{ inconsistent} \]

\[ S_1 \ast S_2 \]
4.3. Inconsistent case

\[ S_1 \cup S_2 \text{ inconsistent} \]

\[ S_1 \star S_2 \]
5. Revision operator and underlying algorithm

5.1. removed set ($r$-set)

Let $R \subset S_1$. $R$ is a $r$-set for the revision operation $S_1 \star S_2$ iff $R$ is the smallest cardinality set such that $(S_1 \setminus R) \cup S_2$ is consistent.
5. Revision operator and underlying algorithm

5.1. removed set (r-set)

Let $R \subset S_1$. $R$ is a r-set for the revision operation $S_1 \ast S_2$ iff $R$ is the smallest cardinality set such that $(S_1 \setminus R) \cup S_2$ is consistent

Answers:
5. Revision operator and underlying algorithm

5.1. removed set (r-set)

Let \( R \subset S_1 \). \( R \) is a r-set for the revision operation \( S_1 \star S_2 \) iff \( R \) is the smallest cardinality set such that \((S_1 \setminus R) \cup S_2\) is consistent

Answers:

- Algorithms (3 solutions studied)
5. Revision operator and underlying algorithm

5.1. removed set (r-set)

Let \( R \subseteq S_1 \). \( R \) is a r-set for the revision operation \( S_1 \star S_2 \) iff \( R \) is the smallest cardinality set such that \( (S_1 \setminus R) \cup S_2 \) is consistent.

Answers:

- Algorithms (3 solutions studied)
- Equivalence (previous work)
5. Revision operator and underlying algorithm

5.1. removed set (r-set)

Let $R \subset S_1$. $R$ is a r-set for the revision operation $S_1 \star S_2$ iff $R$ is the smallest cardinality set such that $(S_1 \setminus R) \cup S_2$ is consistent

Answers:

- Algorithms (3 solutions studied)
- Equivalence (previous work)
- Properties (previous work)
5. Revision operator and underlying algorithm

5.1. removed set (r-set)

Let $R \subset S_1$. $R$ is a r-set for the revision operation $S_1 \star S_2$ iff $R$ is the smallest cardinality set such that $(S_1 \setminus R) \cup S_2$ is consistent.

Answers:

- Algorithms (3 solutions studied)
- Equivalence (previous work)
- Properties (previous work)
- Experimental validation
6. Performances limitations

6.1. Comparing 3 algorithms

![Graph comparing 3 algorithms: REM, MPL, ROBDD]

- **REM**
- **MPL**
- **ROBDD**

**time (s)**

**number of compartments**

3 4 5 6 7 8

300 400 500 600 700 800
6.2. REMr algorithm: pushing to the limits

![Graph showing the relationship between time (s) and the number of compartments for the REMr algorithm.](image-url)
7. Divide and revise

7.1. General ideas
7. Divide and revise

7.1. General ideas

![Diagram with nodes A, B, C, D, E, F and a packet labeled $P_1$]
7. **Divide and revise**

7.1. **General ideas**

![Diagram showing nodes A, B, C, D, E, F with connections and labeled packets P1 and P2.](chart.png)
7. Divide and revise

7.1. General ideas

\[ P = P_1 \cup P_2 \]

Diagram:

- \( P_1 \) packet
- \( P_2 \) packet
7. Divide and revise

7.1. General ideas

\[ P = P_1 \cup P_2 \]
7.2. Example

\[ R(P) = \{ \{E^+, C^-, F^+\}, \{D^+\} \} \]
7.2. Example

\[ P = P_1 \cup P_2 \]

\[ R(P) = \{\{E^+, C^-, F^+\}, \{D^+\}\} \]

- \( R(P_1) = \{\} \) (\( P_1 \) is consistent)
- \( R(P_2) = \{\{E^+\}, \{D^+\}\} \)
7.2. Example

\[ P = P_1 \cup P_2 \]

\[ \mathcal{R}(P) = \{\{E^+, C^-, F^+\}, \{D^+\}\} \]

- \( \mathcal{R}(P_1) = \{\} \) (\( P_1 \) is consistent)
- \( \mathcal{R}(P_2) = \{\{E^+\}, \{D^+\}\} \)

\[ \mathcal{R}(P - \{E^+\}) = \{\{C^-, F^+\}, \{D^+\}\} \quad \mathcal{R}(P - \{D^+\}) = \{\} \]
7.2. Example

\[ R(P) = \{\{E^+, C^-, F^+\}, \{D^+\}\} \]

- \( R(P_1) = \{} \) (\( P_1 \) is consistent)
- \( R(P_2) = \{\{E^+\}, \{D^+\}\} \)

<table>
<thead>
<tr>
<th>( R(P - {E^+}) )</th>
<th>( R(P - {D^+}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>{{C^-, F^+}, {D^+}}</td>
<td>{}</td>
</tr>
<tr>
<td>{E^+} \cup {C^-, F^+}</td>
<td>{D^+}</td>
</tr>
<tr>
<td>{E^+} \cup {D^+} not minimal</td>
<td>{D^+}</td>
</tr>
</tbody>
</table>
7.3. Formal presentation

goal compute removed sets from $P$ starting with $P_1, \ldots, P_n$. 


7.3. **Formal presentation**

**goal** compute removed sets from $P$ starting with $P_1, \ldots, P_n$.

**compute** removed sets $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.
7.3. **Formal presentation**

**goal** compute removed sets from $P$ starting with $P_1, \ldots, P_n$.

**compute** removed sets $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.

**merging** $P_1 \ldots P_n$ : Don’t forget formulas in $P \setminus (P_1 \cup \ldots \cup P_n)$. 
7.3. **Formal presentation**

goal compute removed sets from $P$ starting with $P_1, \ldots, P_n$.

**compute** removed sets $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.

**merging** $P_1 \ldots P_n$: Don’t forget formulas in $P \setminus (P_1 \cup \ldots \cup P_n)$.

\[
(P' \cup P'') = P' \cup P'' \cup \{c \mid c \in KB, \text{lit}(c) \cap (\text{lit}(P') \cup \text{lit}(P'')) \neq \emptyset \}
\]
7.3. Formal presentation

goal compute removed sets from $P$ starting with $P_1, \ldots, P_n$.
compute removed sets $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.
merging $P_1 \ldots P_n$: Don’t forget formulas in $P \setminus (P_1 \cup \ldots \cup P_n)$.

$$(P' \cup P'') = P' \cup P'' \cup \{c \mid c \in KB, \text{lit}(c) \cap (\text{lit}(P') \cup \text{lit}(P'')) \neq \emptyset\}$$

generate every combination of $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.
7.3. Formal presentation

goal compute removed sets from $P$ starting with $P_1, \ldots, P_n$.

compute removed sets $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.

merging $P_1 \ldots P_n$ : Don’t forget formulas in $P \setminus (P_1 \cup \ldots \cup P_n)$.

\[(P' \cup P'') = P' \cup P'' \cup \{c | c \in KB, \text{lit}(c) \cap (\text{lit}(P') \cup \text{lit}(P'')) \neq \emptyset\}\]

generate every combination of $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.

\[\mathcal{C}_1 \otimes \mathcal{C}_2 = \{S | S = S_1 \cup S_2, (S_1, S_2) \in \mathcal{C}_1 \times \mathcal{C}_2\}\]
7.3. Formal presentation

**goal** compute removed sets from $P$ starting with $P_1, \ldots, P_n$.

**compute** removed sets $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.

**merging** $P_1 \ldots P_n$: Don’t forget formulas in $P \setminus (P_1 \cup \ldots \cup P_n)$.

$$\left( P' \cup P'' \right) = P' \cup P'' \cup \{ c \mid c \in KB, \text{lit}(c) \cap (\text{lit}(P') \cup \text{lit}(P'')) \neq \emptyset \}$$

**generate** every combination of $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.

$$\mathcal{C}_1 \otimes \mathcal{C}_2 = \{ S \mid S = S_1 \cup S_2, \ (S_1, S_2) \in \mathcal{C}_1 \times \mathcal{C}_2 \}$$

**compute** removed sets for each $R_i \in \mathcal{R}(P_1) \otimes \ldots \otimes \mathcal{R}(P_n)$, compute the kernels $\mathcal{R}(P \setminus R_i)$. 
7.3. Formal presentation

**goal** compute removed sets from $P$ starting with $P_1, \ldots, P_n$.

**compute** removed sets $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.

**merging** $P_1 \ldots P_n$: Don’t forget formulas in $P \setminus (P_1 \cup \ldots \cup P_n)$.

\[
(P' \cup P'') = P' \cup P'' \cup \{c \mid c \in KB, \text{lit}(c) \cap (\text{lit}(P') \cup \text{lit}(P'')) \neq \emptyset\}
\]

**generate** every combination of $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.

\[
\mathcal{C}_1 \otimes \mathcal{C}_2 = \{S \mid S = S_1 \cup S_2, (S_1, S_2) \in \mathcal{C}_1 \times \mathcal{C}_2\}
\]

**compute** removed sets for each $R_i \in \mathcal{R}(P_1) \otimes \ldots \otimes \mathcal{R}(P_n)$, compute the kernels $\mathcal{R}(P \setminus R_i)$.

**generate** every union of $R_i$ with elements of $\mathcal{R}(P \setminus R_i)$. 
7.3. Formal presentation

goal compute removed sets from $P$ starting with $P_1, \ldots, P_n$.

compute removed sets $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.

merging $P_1 \ldots P_n$ : Don’t forget formulas in $P \setminus (P_1 \cup \ldots \cup P_n)$.

\[
(P' \cup P'') = P' \cup P'' \cup \{c \mid c \in KB, \text{lit}(c) \cap (\text{lit}(P') \cup \text{lit}(P'')) \neq \emptyset\}
\]

generate every combination of $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.

\[
\mathcal{C}_1 \otimes \mathcal{C}_2 = \{S \mid S = S_1 \cup S_2, (S_1, S_2) \in \mathcal{C}_1 \times \mathcal{C}_2\}
\]

compute removed sets for each $R_i \in \mathcal{R}(P_1) \otimes \ldots \otimes \mathcal{R}(P_n)$,
 compute the kernels $\mathcal{R}(P \setminus R_i)$.

generate every union of $R_i$ with elements of $\mathcal{R}(P \setminus R_i)$.

\[
E \bullet \mathcal{C} = \{S \mid S = E \cup C, C \in \mathcal{C}\}
\]
7.3. Formal presentation

goal compute removed sets from $P$ starting with $P_1, \ldots, P_n$.

compute removed sets $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.

merging $P_1 \ldots P_n$ : Don’t forget formulas in $P \setminus (P_1 \cup \ldots \cup P_n)$.

$$ (P' \cup P'') = P' \cup P'' \cup \{ c | c \in KB, \text{lit}(c) \cap (\text{lit}(P') \cup \text{lit}(P'')) \neq \emptyset \} $$

generate every combination of $\mathcal{R}(P_1), \ldots, \mathcal{R}(P_n)$.

$$ \mathcal{C}_1 \otimes \mathcal{C}_2 = \{ S | S = S_1 \cup S_2, (S_1, S_2) \in \mathcal{C}_1 \times \mathcal{C}_2 \} $$

compute removed sets for each $R_i \in \mathcal{R}(P_1) \otimes \ldots \otimes \mathcal{R}(P_n)$, compute the kernels $\mathcal{R}(P \setminus R_i)$.

generate every union of $R_i$ with elements of $\mathcal{R}(P \setminus R_i)$.

$$ E \bullet \mathcal{C} = \{ S | S = E \cup C, C \in \mathcal{C} \} $$

keep only minimal elements of last operation.
7.4. Success condition
7.4. Success condition
7.4. Success condition
7.4. Success condition
8. First results

A few...
8. First results

A few...

Example: 12 compartments case, using a “not up to date” revision algorithm:

- without partitioning: 1200 s
- with partitioning: 400 s
9. Conclusion

- An efficient algorithm for revision operations: REMR
9. **Conclusion**

- An efficient algorithm for revision operations: REMr
- ... But with some limitations.
9. Conclusion

- An efficient algorithm for revision operations: REMr
- ... But with some limitations.
- The “divide and revise” method seem well suited for geographical problems:
9. Conclusion

- An efficient algorithm for revision operations: REMr
- ... But with some limitations.
- The “divide and revise” method seems well suited for geographical problems:
  - We can handle much larger data sets.
9. Conclusion

• An efficient algorithm for revision operations: REMr
• ... But with some limitations.
• The “divide and revise” method seem well suited for geographical problems:
  – We can handle much larger data sets.
  – Conflicts on data are often spatially localized
9. Conclusion

- An efficient algorithm for revision operations: REMR
- ... But with some limitations.
- The “divide and revise” method seem well suited for geographical problems:
  - We can handle much larger data sets.
  - Conflicts on data are often spatially localized.
- We need further testing.
9. Conclusion

• An efficient algorithm for revision operations: REM

• ... But with some limitations.

• The “divide and revise” method seem well suited for geographical problems:
  – We can handle much larger data sets.
  – Conflicts on data are often spatially localized.

• We need further testing.

• We need to evaluate more formally the gains of such a method.