UNIVERSITÀ DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

Preferred Database Repairs under Aggregate Constraints

Sergio Flesca, Filippo Furfaro and Francesco Parisi

D.E.I.S.
University of Calabria

{flesca, furfaro, fparisi}@deis.unical.it

International Conference on Scalable Uncertainty Management (SUM)
Oct 10-12, 2007, Washington DC Area

Inconsistent Numerical databases

- Data inconsistency can arise in several scenarios
  - Data integration, reconciliation,
  - errors in acquiring data (mistakes in transcription, OCR tools, sensors, etc.) → Balance sheet context

A cash budget portion

Receipts Year 2006	cash sales	2200
	receivables	250
	total cash	*2450*

OCR tool →

A digitized cash budget

Receipts Year 2006	cash sales	2200
	receivables	650
	total cash	*2450*

- The original data were consistent: ***2200 + 250 = 2450***, but a symbol recognition error occurred during the digitizing phase
- In this context "*traditional*" forms of constraint do not suffice to guarantee consistency → Aggregate Constraints

Repairing numerical data

- Several consistent versions can be obtained starting from the inconsistent cash budget

A digitized cash budget

Receipts Year 2006	cash sales	2200
	receivables	650
	total cash	*2450*

Repair

Receipts Year 2006	cash sales	X
	receivables	Y
	total cash	*Z*

X, Y, Z such that X+Y=Z

- Some repairs are more reasonable than others
- Card-minimal Repair:
  - A “minimal way” for restoring consistency in databases

    change the minimum number of original values

Card-minimal Repairs

- Several consistent versions can be obtained starting from the inconsistent cash budget

A digitized cash budget

Receipts Year 2006	cash sales	2200
	receivables	650
	total cash	*2450*

Repair →

R_1

Receipts Year 2006	cash sales	X=1800
	receivables	Y=650
	total cash	*Z=2450*

X, Y, Z such that X+Y=Z

- Some repairs are more reasonable than others
- Card-minimal Repair:
  - A "minimal way" for restoring consistency in databases

    → change the minimum number of original values

UNIVERSITÀ DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

Card-minimal Repairs

- Several consistent versions can be obtained starting from the inconsistent cash budget

A digitized cash budget

Receipts Year 2006	cash sales	2200
	receivables	650
	total cash	*2450*

Repair →

R_2

Receipts Year 2006	cash sales	X=2200
	receivables	Y=250
	total cash	*Z=2450*

X, Y, Z such that X+Y=Z

- Some repairs are more reasonable than others
- Card-minimal Repair:
  - A "minimal way" for restoring consistency in databases

    → change the minimum number of original values

Card-minimal Repairs

- Several consistent versions can be obtained starting from the inconsistent cash budget

A digitized cash budget

Receipts Year 2006	cash sales	2200
	receivables	650
	total cash	*2450*

Repair

R_3

Receipts Year 2006	cash sales	X=2200
	receivables	Y=650
	total cash	*Z=2850*

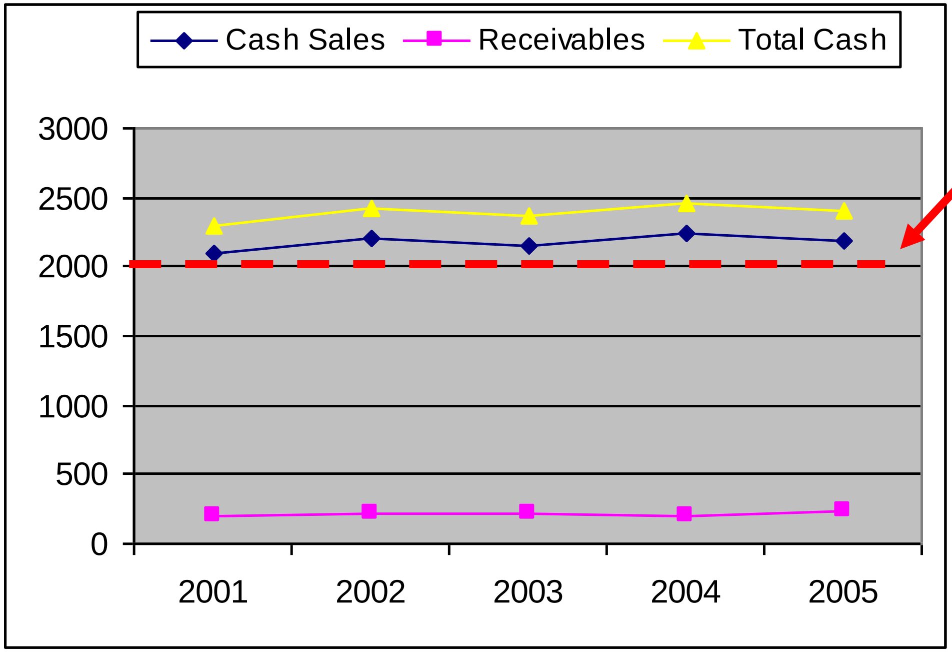
X, Y, Z such that X+Y=Z

- Some repairs are more reasonable than others
- Card-minimal Repair:
  – A "minimal way" for restoring consistency in databases

    change the minimum number of original values

Preferred Repairs

- In general, there may be several card-minimal repairs for a database violating a given set of aggregate constraints
- Well-established information on the application context can be exploited to choose the most reasonable repairs among those having minimum cardinality
  - We can exploit data regarding the preceding years



The value of cash sales never was less than 2000

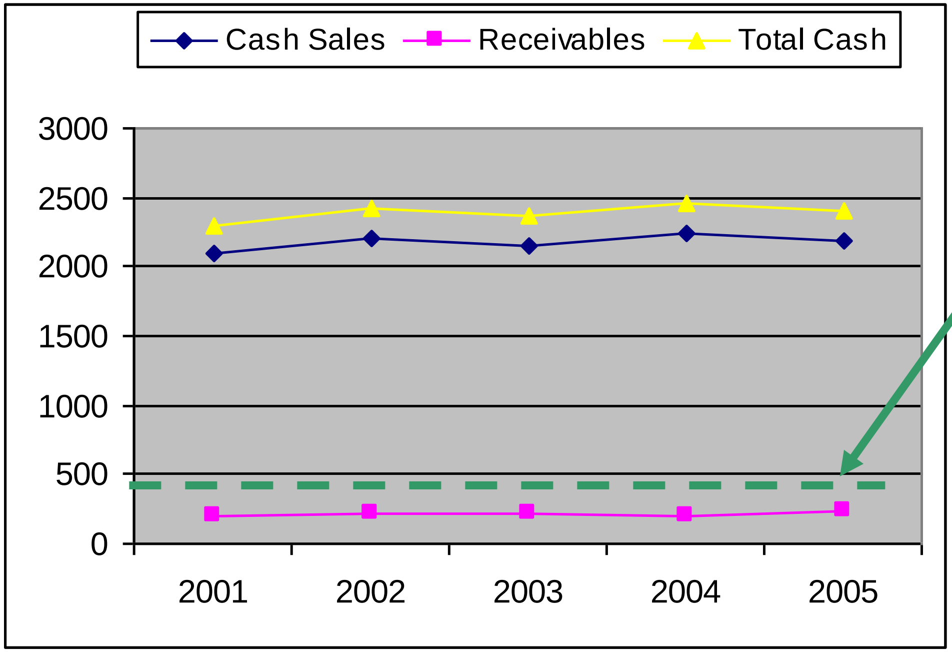
The value of cash sales for the year 2006 is not likely to be less than 2000

This condition can be interpreted as

weak constraint

Preferred Repairs

- In general, there may be several card-minimal repairs for a database violating a given set of aggregate constraints
- Well-established information on the application context can be exploited to choose the most reasonable repairs among those having minimum cardinality
  - We can exploit data regarding the preceding years



The value of receivables never was greater than 400

Weak constraint:

It is likely that receivables are less than or equal to 400

Preferred Repairs

- In general, there may be several card-minimal repairs for a database violating a given set of aggregate constraints
- Well-established information on the application context can be exploited to choose the most reasonable repairs among those having minimum cardinality
  - We can exploit data regarding the preceding years

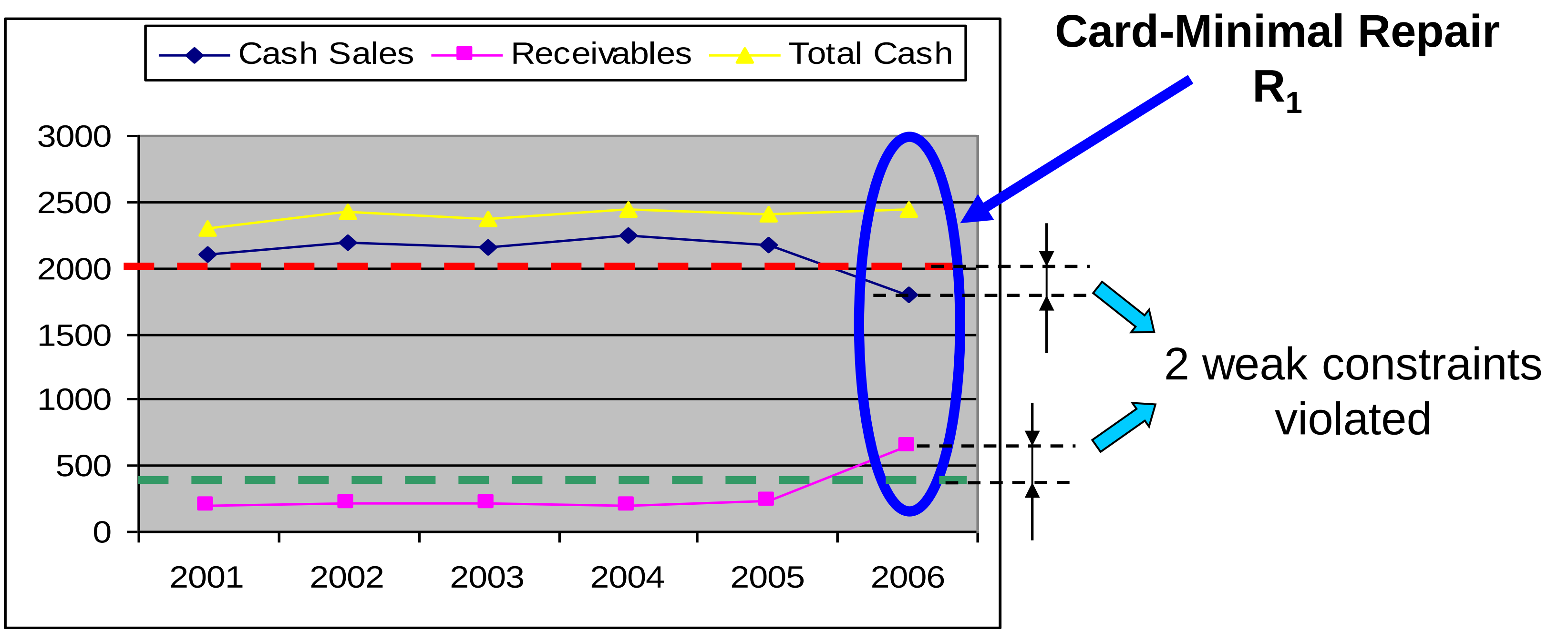
- In contrast with (strong) aggregate constraints, the satisfaction of weak constraints is not mandatory
- Weak constraints can be exploited for defining a repairing technique where inconsistent data are fixed in the "most likely" way

The preferred repairs are card-minimal repairs satisfying as many weak constraints as possible

UNIVERSITÀ DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

Preferred Repairs

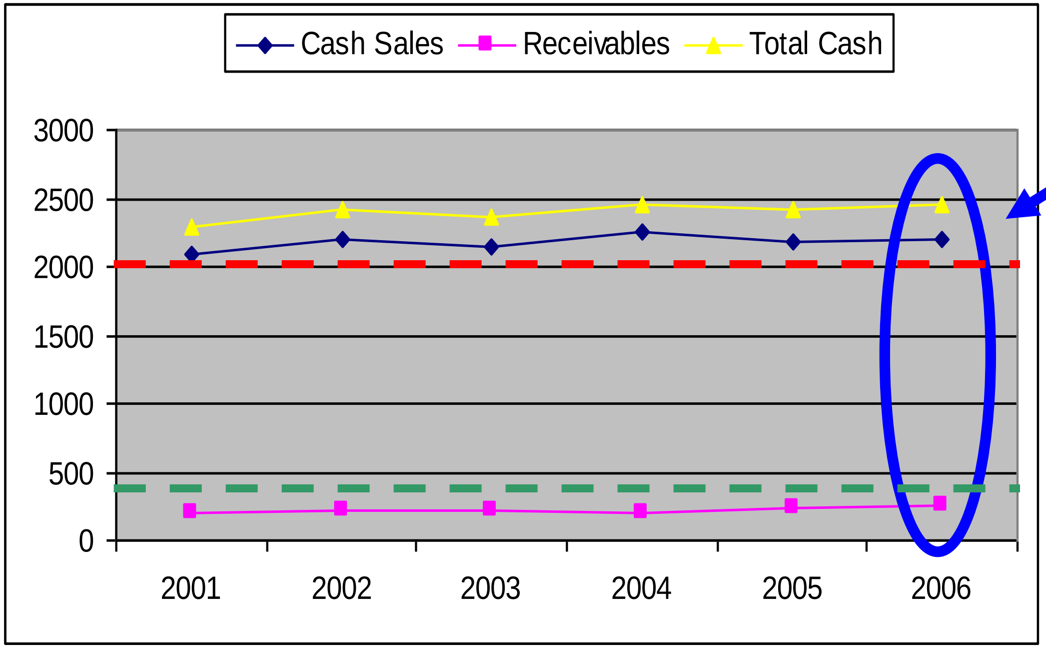
- In general, there may be several card-minimal repairs for a database violating a given set of aggregate constraints
- Well-established information on the application context can be exploited to choose the most reasonable repairs among those having minimum cardinality
  - We can exploit data regarding the preceding years



UNIVERSITÀ DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

Preferred Repairs

- In general, there may be several card-minimal repairs for a database violating a given set of aggregate constraints
- Well-established information on the application context can be exploited to choose the most reasonable repairs among those having minimum cardinality
  - We can exploit data regarding the preceding years



Card-Minimal Repair R_2

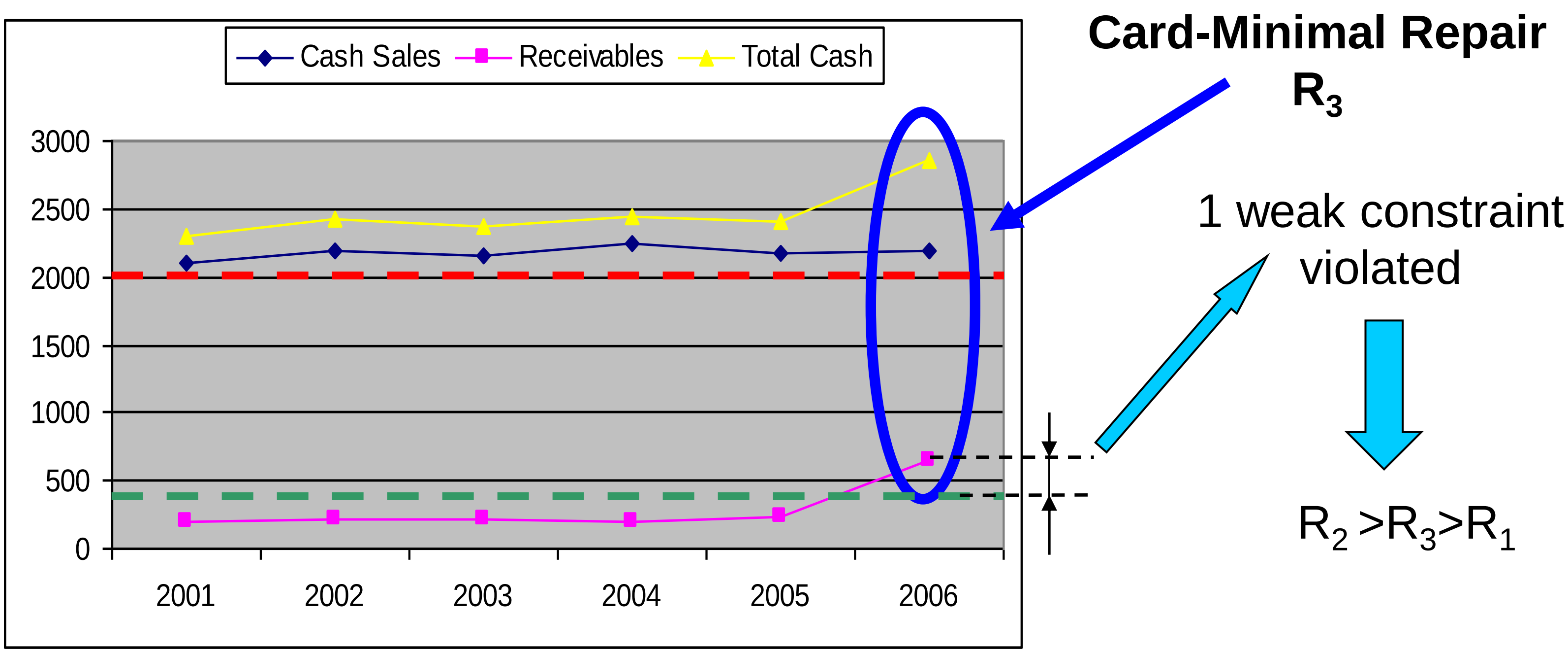
no weak constraints violated

R_2 is preferred to R_1 ($R_2 > R_1$)

UNIVERSITÀ DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

Preferred Repairs

- In general, there may be several card-minimal repairs for a database violating a given set of aggregate constraints
- Well-established information on the application context can be exploited to choose the most reasonable repairs among those having minimum cardinality
  - We can exploit data regarding the preceding years



Outline

- Aggregate constraints
- Repairing strategy
- Weak Aggregate Constraints
- Preferred Repairs
- Steady aggregate constraints
- Complexity results
- Computing preferred repairs
- Experimental results
- Conclusions

Aggregate constraints

- can express constraints like those defined in the context of balance-sheet data

$$\forall\, \overline{X}\ \left(\phi(\overline{X}) \Longrightarrow AggrF(\overline{X}) \leq K\right)$$

where:

1. $\phi(\overline{X})$ is a conjunction of atoms
2. K is a constant
3. The aggregation formula $AggrF(\overline{X})$ is the linear combination of aggregation functions

Linear combination of attributes

$$\chi_i(\overline{X}_i) = SELECT\ sum(e)\ \ FROM\ R\ \ WHERE\ \alpha(\overline{X}_i)$$

Boolean formula on constants and attributes of R

with $\overline{X}_i \subseteq \overline{X}$

Example of aggregate constraints

- CashBudget(Section,Subsection,Type,Value)

Section	Subsection	Type	Value
Receipts	beginning cash	*drv*	*3000*
Receipts	cash sales	*det*	*2200*
Receipts	receivables	*det*	*650*
Receipts	total cash receipts	*aggr*	*2450*
Disbursements	payment of accounts	*det*	*1300*
Disbursements	capital expenditure	*det*	*100*
Disbursements	long-term financing	*det*	*600*
Disbursements	total disbursements	*aggr*	*1000*
Balance	net cash inflow	*drv*	*450*
Balance	ending cash balance	*drv*	*3450*

1)

for each section, the sum of all detail items must be equal to the value of the aggregate item

Aggregation function:

$$\chi_1(s,t) = SELECT \;\; sum(Value) \\ FROM \;\; CashBudget \\ WHERE \;\; Section = s \\ AND \;\; Type = t$$

Aggregate constraint:

$$CashBudget(s,\_,\_,\_) \implies \chi_1(s, det) - \chi_1(s, aggr) = 0$$

Example of aggregate constraints

- CashBudget(Section,Subsection,Type,Value)

Section	Subsection	Type	Value
Receipts	beginning cash	*drv*	*3000*
Receipts	cash sales	*det*	*2200*
Receipts	receivables	*det*	*650*
Receipts	total cash receipts	*aggr*	*2450*
Disbursements	payment of accounts	*det*	*1300*
Disbursements	capital expenditure	*det*	*100*
Disbursements	long-term financing	*det*	*600*
Disbursements	total disbursements	*aggr*	*1000*
Balance	net cash inflow	*drv*	*450*
Balance	ending cash balance	*drv*	*3450*

2)

the net cash inflow must be equal to the difference between total cash receipts and total disbursements

Aggregation function:

$$\chi_2(ss) = \begin{array}{ll} SELECT & sum(Value) \\ FROM & CashBudget \\ WHERE & Subsection = ss \end{array}$$

Aggregate constraint:

$$\chi_2(net\ cash\ inflow) - [\chi_2(total\ cash\ receipts) - \chi_2(total\ disbursements)] = 0$$

Example of aggregate constraints

- CashBudget(Section,Subsection,Type,Value)

Section	Subsection	Type	Value
Receipts	beginning cash	*drv*	*3000*
Receipts	cash sales	*det*	*2200*
Receipts	receivables	*det*	*650*
Receipts	total cash receipts	*aggr*	*2450*
Disbursements	payment of accounts	*det*	*1300*
Disbursements	capital expenditure	*det*	*100*
Disbursements	long-term financing	*det*	*600*
Disbursements	total disbursements	*aggr*	*1000*
Balance	net cash inflow	*drv*	*450*
Balance	ending cash balance	*drv*	*3450*

3)

the ending cash balance must be equal to the sum of the beginning cash and the net cash inflow

Aggregation function:

$$\chi_2(ss) = SELECT \ \ sum(Value) \\ FROM \ \ CashBudget \\ WHERE \ \ Subsection = ss$$

Aggregate constraint:

$$\chi_2(ending\ cash\ balance) - [\chi_2(beginning\ cash) + \chi_2(net\ cash\ balance)] = 0$$

Outline

- Aggregate constraints
- Repairing strategy
- Weak Aggregate Constraints
- Preferred Repairs
- Steady aggregate constraints
- Complexity results
- Computing preferred repairs
- Experimental results
- Conclusions

UNIVERSITÀ DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

Repairing strategy

- What is a reasonable strategy for repairing the acquired data?

~~**Tuple deletion / insertion**~~

The inconsistent cash budget

Receipts	cash sales	2200
	receivables	650
	total cash	***2450***

2200 + 650 ≠ 2450

The repaired cash budget

Receipts	cash sales	2200
	receivables	650
	XXXXX	**-400**
	total cash	***2450***

2200 +
650 -
400 =
2450

Adding a new tuple means that the OCR tool skipped a whole row when acquiring ... ***It's rather unrealistic!!!***

Repairing strategy

- What is a reasonable strategy for repairing the acquired data?
- The most natural approach is updating directly the numerical data
  - Work at attribute-level, rather than tuple-level

The inconsistent cash budget

Receipts	cash sales	2200
	receivables	650
	total cash	***2450***

2200 + 650 ≠ 2450

A repaired cash budget

Receipts	cash sales	2200
	receivables	250
	total cash	***2450***

2200 +
250 =
2450

- In our context, we can reasonably assume that inconsistencies are due to symbol recognition errors
- Thus, trying to re-construct the actual data values (without changing the number of tuples) is well founded

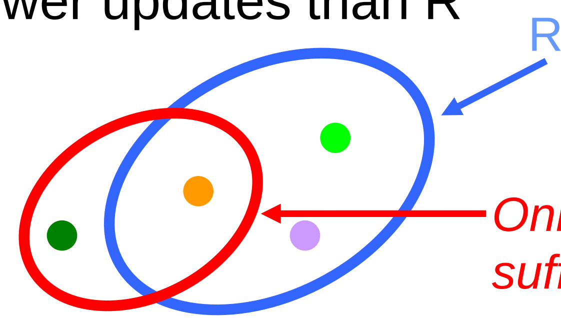
Repairing strategy

- (Minimal) Repair
  - A “minimal way” for restoring consistency in databases

    → preserve as much information as possible

CARD-MINIMAL SEMANTICS

- A repair R is *card*-minimal for D iff there is no repair R’ for D consisting of fewer updates than R



Only two updates do not suffice to repair D!

  - It means assuming that the minimum number of errors occurred
    - ***In the balance-sheet context***: the most probable case is that the acquiring system made the minimum number of errors

Two examples of card-minimal repair

Section	Subsection	Type	Value
Receipts	beginning cash	*drv*	*3000*
Receipts	cash sales	*det*	*2200*
Receipts	receivables	*det*	*650*
Receipts	total cash receipts	*aggr*	*2450*
Disbursements	payment of accounts	*det*	*1300*
Disbursements	capital expenditure	*det*	*100*
Disbursements	long-term financing	*det*	*600*
Disbursements	total disbursements	*aggr*	*1000*
Balance	net cash inflow	*drv*	*450*
Balance	ending cash balance	*drv*	*3450*

R_1: cash sales → **1800**; total disbursements → **2000**

R_2: receivables → **250**; total disbursements → **2000**

Constraint 1) → satisfied

- for each section, the sum of all detail items must be equal to the value of the aggregate item

Two examples of card-minimal repair

Section	Subsection	Type	Value
Receipts	beginning cash	*drv*	*3000*
Receipts	cash sales	*det*	*2200*
Receipts	receivables	*det*	*650*
Receipts	total cash receipts	*aggr*	*2450*
Disbursements	payment of accounts	*det*	*1300*
Disbursements	capital expenditure	*det*	*100*
Disbursements	long-term financing	*det*	*600*
Disbursements	total disbursements	*aggr*	*1000*
Balance	net cash inflow	*drv*	*450*
Balance	ending cash balance	*drv*	*3450*

R_1: cash sales → **1800**; total disbursements → **2000**

R_2: receivables → **250**; total disbursements → **2000**

Constraint 2) → satisfied

- the net cash inflow must be equal to the difference between total cash receipts and total disbursements

Two examples of card-minimal repair

Section	Subsection	Type	Value
Receipts	beginning cash	*drv*	*3000*
Receipts	cash sales	*det*	*2200*
Receipts	receivables	*det*	*650*
Receipts	total cash receipts	*aggr*	*2450*
Disbursements	payment of accounts	*det*	*1300*
Disbursements	capital expenditure	*det*	*100*
Disbursements	long-term financing	*det*	*600*
Disbursements	total disbursements	*aggr*	*1000*
Balance	net cash inflow	*drv*	*450*
Balance	ending cash balance	*drv*	*3450*

R_1: cash sales 2200 → **1800**; total disbursements 1000 → **2000**

R_2: receivables 650 → **250**; total disbursements 1000 → **2000**

Constraint 3) → satisfied

- the ending cash balance must be equal to the sum of the beginning cash and the net cash inflow

Outline

- Aggregate constraints
- Repairing strategy
- Weak Aggregate Constraints
- Preferred Repairs
- Steady aggregate constraints
- Complexity results
- Computing preferred repairs
- Experimental results
- Conclusions

Weak aggregate constraints

- Aggregate constraints with a “weak” semantics

- In contrast with the “strong” semantics of aggregate constraints, weak aggregate constraints express conditions which reasonably hold in the actual data, although satisfying them is not mandatory

- The condition *“it is likely that cash sales are greater than or equal to 2000”* can be expressed by

$$\chi_2(\text{‘cash sales’}) \geq 2000$$

- Whereas, the condition *“it is likely that receivables are less than or equal to 400”* can be expressed by

$$\chi_2(\text{‘receivables’}) \leq 400$$

where: $$\begin{aligned}\chi_2(ss) = &\ SELECT\ \ sum(Value)\\ &\ FROM\ \ \ CashBudget\\ &\ WHERE\ Subsection = ss\end{aligned}$$

Outline

- Aggregate constraints
- Repairing strategy
- Weak Aggregate Constraints
- Preferred Repairs
- Steady aggregate constraints
- Complexity results
- Computing preferred repairs
- Experimental results
- Conclusions

Preferred Repairs

- Card-minimal repairs can be ordered according the number of conditions expressed by the set of weak constraints which are satisfied in the repaired database
- A card-minimal repair violating *n* ground weak constraints is preferred to any other card-minimal repair violating *m>n* ground weak constraints

Section	Subsection	Type	Value
Receipts	beginning cash	*drv*	*3000*
Receipts	cash sales	*det*	*2200*
Receipts	receivables	*det*	*650*
Receipts	total cash receipts	*aggr*	*2450*
Disbursements	payment of accounts	*det*	*1300*
Disbursements	capital expenditure	*det*	*100*
Disbursements	long-term financing	*det*	*600*
Disbursements	total disbursements	*aggr*	*1000*
Balance	net cash inflow	*drv*	*450*
Balance	ending cash balance	*drv*	*3450*

R_1: **1800** (cash sales), **2000** (total disbursements)

R_2: **250** (receivables), **2000** (total disbursements)

R_2 is preferred to R_1

weak constraints: $\chi_2(\text{'cash sales'}) \geq 2000$ $\chi_2(\text{'receivables'}) \leq 400$

Outline

- Aggregate constraints
- Repairing strategy
- Weak Aggregate Constraints
- Preferred Repairs
- Steady aggregate constraints
- Complexity results
- Computing preferred repairs
- Experimental results
- Conclusions

Steady aggregate constraints (SACs)

- A restricted but expressive class of aggregate constraints
  - Computing a preferred repair for a database D w.r.t. a set of steady aggregate constraint AC and a set of steady weak aggregate constraint W can be accomplished by solving an instance of ILP problem

An aggregate constraint is an SAC if:

1) no attributes in the WHERE clause are measure attributes

UNIVERSITÀ DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

Steady aggregate constraints (SACs)

- A restricted but expressive class of aggregate constraints
  - Computing a preferred repair for a database D w.r.t. a set of steady aggregate constraint AC and a set of steady weak aggregate constraint W can be accomplished by solving an instance of ILP problem

An aggregate constraint is an SAC if:

1) no attributes in the WHERE clause are measure attributes

Attributes whose values can be changed by a repair

- CashBudget(Section,Subsection,Type,Value)

$$CashBudget(s, ss, t, v) \implies \chi_1(s, det) - \chi_1(s, aggr) = 0$$

where: $\chi_1(s,t) = \begin{array}{ll} SELECT & sum(Value) \\ FROM & CashBudget \\ WHERE & Section = s \\ \quad AND & Type = t \end{array}$

Steady aggregate constraints (SACs)

- A restricted but expressive class of aggregate constraints
  - Computing a preferred repair for a database D w.r.t. a set of steady aggregate constraint AC and a set of steady weak aggregate constraint W can be accomplished by solving an instance of ILP problem

An aggregate constraint is an SAC if:

1) no attributes in the WHERE clause are measure attributes
2) no attributes corresponding to variables in the WHERE clause are measure attributes

- CashBudget(Section,Subsection,Type,Value)

$$CashBudget(s, ss, t, v) \implies \chi_1(s, det) - \chi_1(s, aggr) = 0$$

where: $\chi_1(s,t) = SELECT\ sum(Value)\ FROM\ CashBudget\ WHERE\ Section = s\ AND\ Type = t$

Steady aggregate constraints (SACs)

- A restricted but expressive class of aggregate constraints
  - Computing a preferred repair for a database D w.r.t. a set of steady aggregate constraint AC and a set of steady weak aggregate constraint W can be accomplished by solving an instance of ILP problem

An aggregate constraint is an SAC if:

1) no attributes in the WHERE clause are measure attributes
2) no attributes corresponding to variables in the WHERE clause are measure attributes
3) no attributes corresponding to variables shared by two atoms are measure attributes

- CashBudget(Section,Subsection,Type,Value)

$$CashBudget(s, ss, t, v) \implies \chi_1(s, det) - \chi_1(s, aggr) = 0$$

where: $\chi_1(s,t) = \begin{array}{ll} SELECT & sum(Value) \\ FROM & CashBudget \\ WHERE & Section = s \\ \quad AND & Type = t \end{array}$

UNIVERSITÀ DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

Outline

- Aggregate constraints
- Repairing strategy
- Weak Aggregate Constraints
- Preferred Repairs
- Steady aggregate constraints
- Complexity results
- Computing preferred repairs
- Experimental results
- Conclusions

Complexity Results

- Given a database D, a set of aggregate constraints AC and a set of weak aggregate constraints W

1) Deciding whether there is a preferred repair for D w.r.t. AC and W violating more than k ground weak constraints is NP-complete
   - The problem is NP-hard even in the case that both AC and W consist of steady constraints only

2) Given a repair R for D w.r.t. AC, deciding whether R is a preferred repair for D w.r.t. AC and W is coNP-complete
   - The problem is coNP-hard even in the case that both AC and W consist of steady constraints only

- Steady constraints do not affect the complexity of the preferred-repair existence problem and of the preferred-repair checking problem

Outline

- Aggregate constraints
- Repairing strategy
- Weak Aggregate Constraints
- Preferred Repairs
- Steady aggregate constraints
- Complexity results
- Computing preferred repairs
- Experimental results
- Conclusions

Computing Preferred Repairs

- Under SACs a preferred repair can be computed solving an ILP problem instance
  1. Strong SACs are translated into a system S of linear inequalities

Section	Subsection	Type	Value
Receipts	beginning cash	*drv*	*3000*
Receipts	cash sales	*det*	*2200*
Receipts	receivables	*det*	*650*
Receipts	total cash receipts	*aggr*	*2450*
Disbursements	payment of accounts	*det*	*1300*
Disbursements	capital expenditure	*det*	*100*
Disbursements	long-term financing	*det*	*600*
Disbursements	total disbursements	*aggr*	*1000*
Balance	net cash inflow	*drv*	*450*
Balance	ending cash balance	*drv*	*3450*

z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8, z_9, z_{10}

$$\begin{cases} z_2 + z_3 = z_4 \\ z_5 + z_6 + z_7 = z_8 \end{cases}$$

$$CashBudget(s, \_, \_, \_) \implies \chi_1(s, det) - \chi_1(s, aggr) = 0$$

Computing Preferred Repairs

- Under SACs a preferred repair can be computed solving an ILP problem instance

1. Strong SACs are translated into a system S of linear inequalities
   - Each solution s of S corresponds to a repair R(s)
   - In general, R(s) is a non-minimal and non-preferred repair
2. Further linear inequalities are added in order to decide whether a solution s of S corresponds to R(s) is a preferred repair

Computing Preferred Repairs

- Under SACs a preferred repair can be computed solving an ILP problem instance

2. Further linear inequalities are added in order to decide whether a solution s of S corresponds to R(s) is a preferred repair

$$\min\left(\sum_{i \in \mathcal{I}_{AC}} N \cdot \delta_i + \sum_{\omega \in gr(\mathcal{W})} \mu_\omega \right)$$

$$\begin{cases} \mathbf{A} \times \mathbf{Z} \leq \mathbf{B} & \\ \boxed{y_i = z_i - v_i \qquad \forall\, i \in \mathcal{I}_{AC}} & \\ y_i \leq M \cdot \delta_i & \forall\, i \in \mathcal{I}_{AC} \\ -M \cdot \delta_i \leq y_i & \forall\, i \in \mathcal{I}_{AC} \\ \sigma_\omega = K_\omega - Q(\omega) & \forall\, \omega \in gr(\mathcal{W}) \\ -M \cdot \mu_\omega \leq \sigma_\omega & \forall\, \omega \in gr(\mathcal{W}) \\ \boxed{\delta_i \in \{0,1\} \qquad \forall\, i \in \mathcal{I}_{AC}} & \\ \mu_\omega \in \{0,1\} & \forall\, \omega \in gr(\mathcal{W}) \end{cases}$$

for each database value v_i we define an integer variable y_i and a binary variable δ_i

Computing Preferred Repairs

- Under SACs a preferred repair can be computed solving an ILP problem instance

2. Further linear inequalities are added in order to decide whether a solution s of S corresponds to R(s) is a preferred repair

$$\min \left(\sum_{i \in \mathcal{I}_{AC}} N \cdot \delta_i + \sum_{\omega \in gr(\mathcal{W})} \mu_\omega \right)$$

$$\begin{cases} \mathbf{A} \times \mathbf{Z} \leq \mathbf{B} & \\ \boxed{y_i = z_i - v_i \qquad \forall\, i \in \mathcal{I}_{AC}} & \\ y_i \leq M \cdot \delta_i & \forall\, i \in \mathcal{I}_{AC} \\ -M \cdot \delta_i \leq y_i & \forall\, i \in \mathcal{I}_{AC} \\ \sigma_\omega = K_\omega - Q(\omega) & \forall\, \omega \in gr(\mathcal{W}) \\ -M \cdot \mu_\omega \leq \sigma_\omega & \forall\, \omega \in gr(\mathcal{W}) \\ \boxed{\delta_i \in \{0,1\} \qquad \forall\, i \in \mathcal{I}_{AC}} & \\ \mu_\omega \in \{0,1\} & \forall\, \omega \in gr(\mathcal{W}) \end{cases}$$

$y_i \neq 0$ ⇨ *database value v_i updated by R(s)*

Computing Preferred Repairs

- Under SACs a preferred repair can be computed solving an ILP problem instance

2. Further linear inequalities are added in order to decide whether a solution s of S corresponds to R(s) is a preferred repair

$$\min\left(\sum_{i\in\mathcal{I}_{AC}} N\cdot\delta_i + \sum_{\omega\in gr(\mathcal{W})}\mu_\omega\right)$$

$$\begin{cases}
\mathbf{A}\times\mathbf{Z}\leq\mathbf{B} & \\
y_i = z_i - v_i & \forall\, i\in\mathcal{I}_{AC}\\
\boxed{\begin{array}{ll} y_i \leq M\cdot\delta_i & \forall\, i\in\mathcal{I}_{AC}\\ -M\cdot\delta_i\leq y_i & \forall\, i\in\mathcal{I}_{AC}\end{array}} & \\
\sigma_\omega = K_\omega - Q(\omega) & \forall\,\omega\in gr(\mathcal{W})\\
-M\cdot\mu_\omega\leq\sigma_\omega & \forall\,\omega\in gr(\mathcal{W})\\
\delta_i\in\{0,1\} & \forall\, i\in\mathcal{I}_{AC}\\
\mu_\omega\in\{0,1\} & \forall\,\omega\in gr(\mathcal{W})
\end{cases}$$

$y_i\neq 0 \Rightarrow$ *database value v_i updated by R(s)*

$y_i\neq 0 \Rightarrow \delta_i=1$

Computing Preferred Repairs

- Under SACs a preferred repair can be computed solving an ILP problem instance

2. Further linear inequalities are added in order to decide whether a solution s of S corresponds to R(s) is a preferred repair

$$\min \left(\sum_{i \in \mathcal{I}_{AC}} N \cdot \delta_i + \sum_{\omega \in gr(\mathcal{W})} \mu_\omega \right)$$

$$\begin{cases} \mathbf{A} \times \mathbf{Z} \leq \mathbf{B} & \\ y_i = z_i - v_i & \forall\, i \in \mathcal{I}_{AC} \\ y_i \leq M \cdot \delta_i & \forall\, i \in \mathcal{I}_{AC} \\ -M \cdot \delta_i \leq y_i & \forall\, i \in \mathcal{I}_{AC} \\ \sigma_\omega = K_\omega - Q(\omega) & \forall\, \omega \in gr(\mathcal{W}) \\ -M \cdot \mu_\omega \leq \sigma_\omega & \forall\, \omega \in gr(\mathcal{W}) \\ \delta_i \in \{0,1\} & \forall\, i \in \mathcal{I}_{AC} \\ \mu_\omega \in \{0,1\} & \forall\, \omega \in gr(\mathcal{W}) \end{cases}$$

$y_i \neq 0 \Rightarrow$ *database value v_i updated by R(s)*

$y_i \neq 0 \Rightarrow \delta_i = 1$

If a system of equalities has a solution, it has also one where each variable takes a value in [-M,M]

Computing Preferred Repairs

- Under SACs a preferred repair can be computed solving an ILP problem instance

2. Further linear inequalities are added in order to decide whether a solution s of S corresponds to R(s) is a preferred repair

$$\boxed{\min \left(\sum_{i \in \mathcal{I}_{AC}} N \cdot \delta_i \right.} + \left. \sum_{\omega \in gr(\mathcal{W})} \mu_\omega \right)$$

$$\begin{cases} \mathbf{A} \times \mathbf{Z} \leq \mathbf{B} & \\ y_i = z_i - v_i & \forall\, i \in \mathcal{I}_{AC} \\ y_i \leq M \cdot \delta_i & \forall\, i \in \mathcal{I}_{AC} \\ -M \cdot \delta_i \leq y_i & \forall\, i \in \mathcal{I}_{AC} \\ \sigma_\omega = K_\omega - Q(\omega) & \forall\, \omega \in gr(\mathcal{W}) \\ -M \cdot \mu_\omega \leq \sigma_\omega & \forall\, \omega \in gr(\mathcal{W}) \\ \delta_i \in \{0,1\} & \forall\, i \in \mathcal{I}_{AC} \\ \mu_\omega \in \{0,1\} & \forall\, \omega \in gr(\mathcal{W}) \end{cases}$$

$y_i \neq 0 \Rightarrow$ *database value v_i updated by R(s)*

$y_i \neq 0 \Rightarrow \delta_i = 1$

minimizing the sum of values assigned to the binary variables δ_i means searching for card-minimal repairs

Computing Preferred Repairs

- Under SACs a preferred repair can be computed solving an ILP problem instance

2. Further linear inequalities are added in order to decide whether a solution s of S corresponds to R(s) is a preferred repair

$$\min\left(\sum_{i\in\mathcal{I}_{AC}} N\cdot\delta_i + \sum_{\omega\in gr(\mathcal{W})}\mu_\omega \right)$$

$$\begin{cases} \mathbf{A}\times\mathbf{Z}\leq\mathbf{B} & \\ y_i = z_i - v_i & \forall i\in\mathcal{I}_{AC} \\ y_i \leq M\cdot\delta_i & \forall i\in\mathcal{I}_{AC} \\ -M\cdot\delta_i\leq y_i & \forall i\in\mathcal{I}_{AC} \\ \boxed{\sigma_\omega = K_\omega - Q(\omega) \quad \forall\,\omega\in gr(\mathcal{W})} & \\ -M\cdot\mu_\omega\leq\sigma_\omega & \forall\,\omega\in gr(\mathcal{W}) \\ \delta_i\in\{0,1\} & \forall i\in\mathcal{I}_{AC} \\ \boxed{\mu_\omega\in\{0,1\} \quad \forall\,\omega\in gr(\mathcal{W})} & \end{cases}$$

for each ground weak constraint ω we define a variable σ_ω and a binary variable μ_ω

σ_ω *< 0 means constraint* ω violated

$\omega = \chi_2(\text{'cash sales'}) \geq 2000$

Section	Subsection	Type	Value
…	…	…	…
Receipts	cash sales	*det*	*2200*
…	…	…	…

$\rightarrow z_2$

$\sigma_\omega = 2000 - z_2$

Computing Preferred Repairs

- Under SACs a preferred repair can be computed solving an ILP problem instance

2. Further linear inequalities are added in order to decide whether a solution s of S corresponds to R(s) is a preferred repair

$$\min\left(\sum_{i\in\mathcal{I}_{AC}} N\cdot\delta_i + \sum_{\omega\in gr(\mathcal{W})}\mu_\omega\right)$$

$$\begin{cases}
\mathbf{A}\times\mathbf{Z}\leq\mathbf{B} & \\
y_i = z_i - v_i & \forall\, i\in\mathcal{I}_{AC}\\
y_i \leq M\cdot\delta_i & \forall\, i\in\mathcal{I}_{AC}\\
-M\cdot\delta_i\leq y_i & \forall\, i\in\mathcal{I}_{AC}\\
\sigma_\omega = K_\omega - Q(\omega) & \forall\,\omega\in gr(\mathcal{W})\\
\boxed{-M\cdot\mu_\omega\leq\sigma_\omega \qquad \forall\,\omega\in gr(\mathcal{W})} & \\
\delta_i\in\{0,1\} & \forall\, i\in\mathcal{I}_{AC}\\
\mu_\omega\in\{0,1\} & \forall\,\omega\in gr(\mathcal{W})
\end{cases}$$

for each ground constraint ω we define a variable σ_ω and a binary variable μ_ω

$\sigma_\omega < 0$ *means constraint* ω violated

$\sigma_\omega < 0 \Rightarrow \mu_\omega = 1$

UNIVERSITÀ DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

Computing Preferred Repairs

- Under SACs a preferred repair can be computed solving an ILP problem instance

2. Further linear inequalities are added in order to decide whether a solution s of S corresponds to R(s) is a preferred repair

$$\min \left(\sum_{i \in \mathcal{I}_{AC}} N \cdot \delta_i + \boxed{\sum_{\omega \in gr(\mathcal{W})} \mu_\omega} \right)$$

$$\begin{cases} \mathbf{A} \times \mathbf{Z} \leq \mathbf{B} & \\ y_i = z_i - v_i & \forall i \in \mathcal{I}_{AC} \\ y_i \leq M \cdot \delta_i & \forall i \in \mathcal{I}_{AC} \\ -M \cdot \delta_i \leq y_i & \forall i \in \mathcal{I}_{AC} \\ \sigma_\omega = K_\omega - Q(\omega) & \forall \omega \in gr(\mathcal{W}) \\ -M \cdot \mu_\omega \leq \sigma_\omega & \forall \omega \in gr(\mathcal{W}) \\ \delta_i \in \{0,1\} & \forall i \in \mathcal{I}_{AC} \\ \mu_\omega \in \{0,1\} & \forall \omega \in gr(\mathcal{W}) \end{cases}$$

for each ground constraint ω we define a variable σ_ω and a binary variable μ_ω

$\sigma_\omega < 0$ *means constraint* ω violated

$\sigma_\omega < 0 \Rightarrow \mu_\omega = 1$

minimizing the sum of values assigned to the binary variables μ_ω means searching for card-minimal repairs violating as few weak constraints as possible

UNIVERSITÀ DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

Computing Preferred Repairs

- Under SACs a preferred repair can be computed solving an ILP problem instance

$$\min \left(\sum_{i \in \mathcal{I}_{AC}} N \cdot \delta_i + \sum_{\omega \in gr(\mathcal{W})} \mu_\omega \right)$$

$$\begin{cases} \mathbf{A} \times \mathbf{Z} \leq \mathbf{B} & \\ y_i = z_i - v_i & \forall\, i \in \mathcal{I}_{AC} \\ y_i \leq M \cdot \delta_i & \forall\, i \in \mathcal{I}_{AC} \\ -M \cdot \delta_i \leq y_i & \forall\, i \in \mathcal{I}_{AC} \\ \sigma_\omega = K_\omega - Q(\omega) & \forall\, \omega \in gr(\mathcal{W}) \\ -M \cdot \mu_\omega \leq \sigma_\omega & \forall\, \omega \in gr(\mathcal{W}) \\ \delta_i \in \{0,1\} & \forall\, i \in \mathcal{I}_{AC} \\ \mu_\omega \in \{0,1\} & \forall\, \omega \in gr(\mathcal{W}) \end{cases}$$

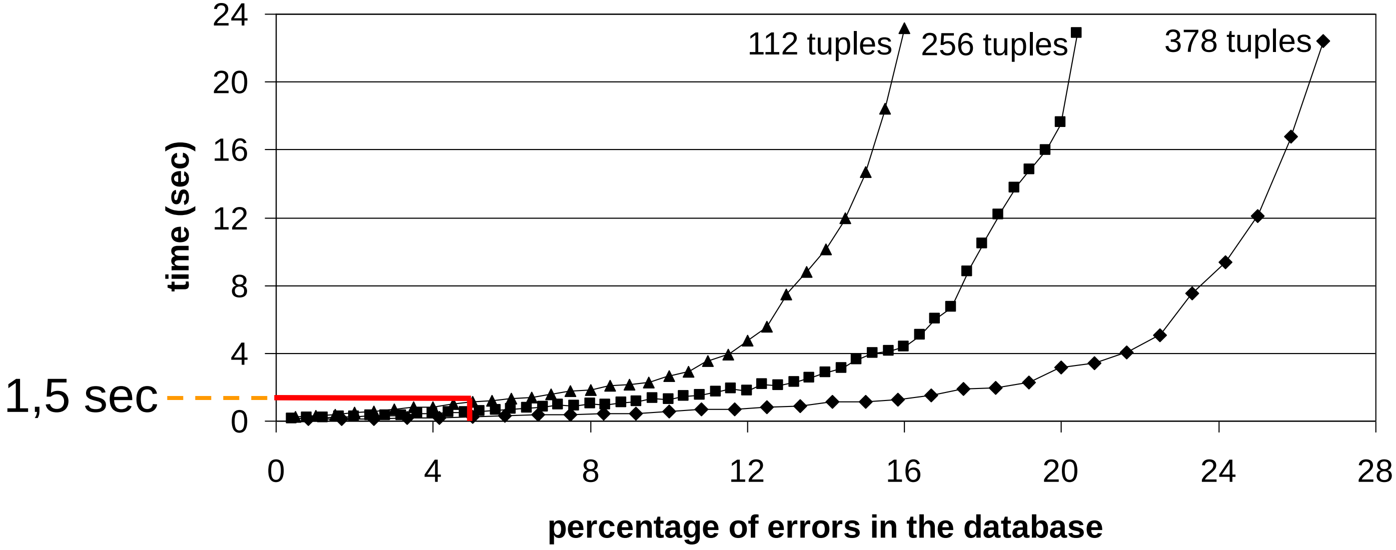
every optimal solution of this problem corresponds to an M-bounded preferred repair and vice versa

Outline

- Aggregate constraints
- Repairing strategy
- Weak Aggregate Constraints
- Preferred Repairs
- Steady aggregate constraints
- Complexity results
- Computing preferred repairs
- Experimental results
- Conclusions

Experimental Results

- Application context: balance-sheet data
  - the number of item occurring in a balance-sheet is unlikely to be greater than 400
  - the percentage of erroneous items is less than 5% of the acquired data
- Time employed for computing a *preferred* repair

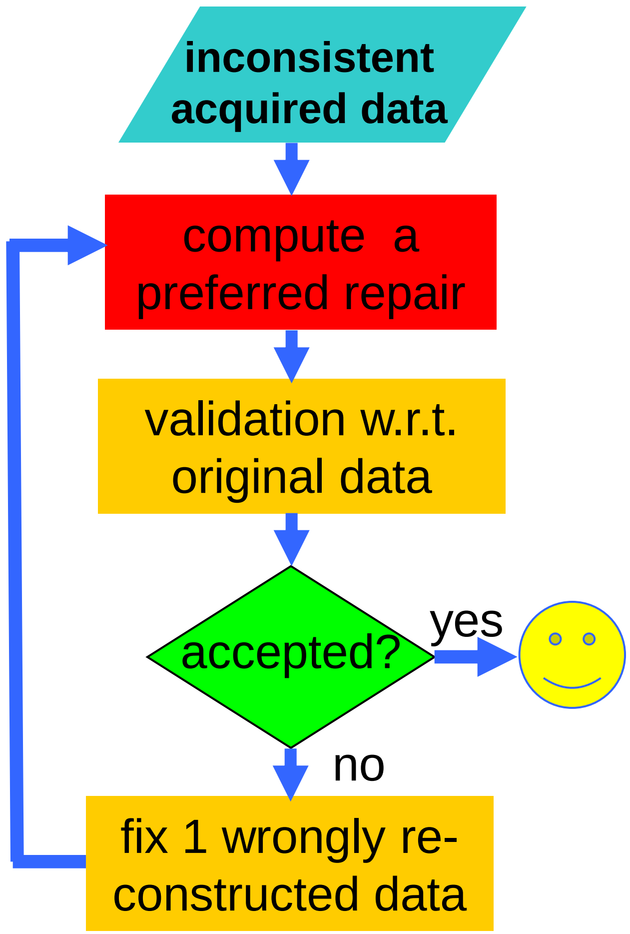


- The technique can be effectively employed in the balance-sheet context

UNIVERSITÀ DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

Experimental Results

- The prototype can be used in a semi-automatic system for fixing data acquisition errors



Impact of using weak constraints

Average number of iteration for re-construct the original data

percentage of errors in the DB	*1%*	*2%*	*4%*	*6%*
without	2.8	4.4	6	7.1
with weak constraints	1.2	1.5	1.9	2.3

Conclusions

- A framework for computing preferred repairs in numerical data violating a given set of strong and weak aggregate constraints has been proposed
- The proposed approach exploits a transformation of the problem of computing a preferred repair into an instance of ILP problem
  - standard techniques addressing ILP problem can be re-used for computing a preferred repair
- The prototype can be used in a semi-automatic system for fixing data acquisition errors
  - Experimental results prove the effectiveness in the balance-sheet context

Thank you!

...*any questions?*