Frequent Generalized Subgraph Mining via Graph Edit Distances

Richard Palme and Pascal Welke

IoT Streams for Predictive Maintenance | SeDaMi

UNIVERSITÄT BONN

A Motivating Use Case

A Motivating Use Case

A Predictive Maintenance Scenario

- Farmers track their livestock and have to treat them well

A Predictive Maintenance Scenario

- Farmers track their livestock and have to treat them well
  - before they get sick

A Predictive Maintenance Scenario

- Farmers track their livestock and have to treat them well
  - before they get sick
  - certainly before they die

A Predictive Maintenance Scenario

- Farmers track their livestock and have to treat them well
  - before they get sick
  - certainly before they die
  - so that they perform well for a long time

A Predictive Maintenance Scenario

- Farmers track their livestock and have to treat them well
  - before they get sick
  - certainly before they die
  - so that they perform well for a long time
- Breeders track whole populations of cows and need to decide

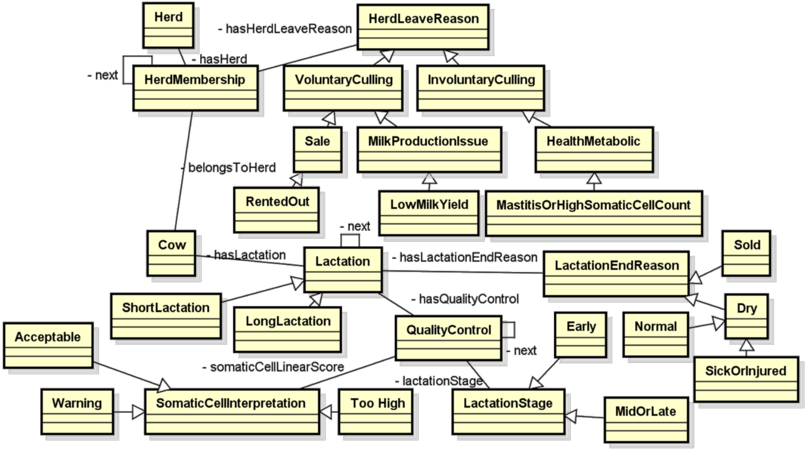
A Predictive Maintenance Scenario

- Farmers track their livestock and have to treat them well
  - before they get sick
  - certainly before they die
  - so that they perform well for a long time
- Breeders track whole populations of cows and need to decide
  - if a breed is a good fit for a changing market or changing climate

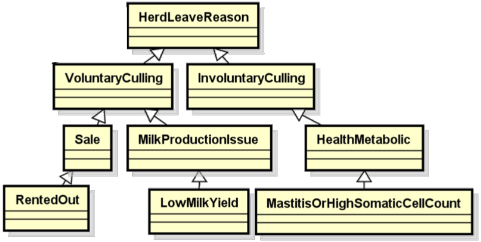
A Predictive Maintenance Scenario

- Farmers track their livestock and have to treat them well
  - before they get sick
  - certainly before they die
  - so that they perform well for a long time
- Breeders track whole populations of cows and need to decide
  - if a breed is a good fit for a changing market or changing climate
  - which traits to improve by selective breeding

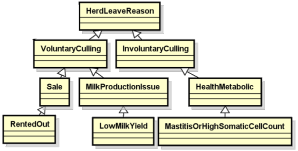
A Dairy Cattle Performance Ontology



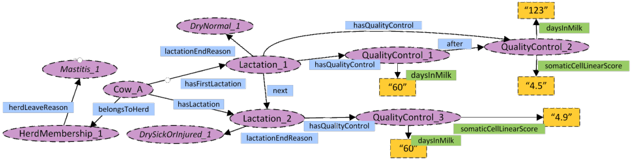
A Dairy Cattle Performance Ontology

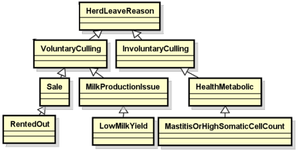


Dairy Cattle Performance Data

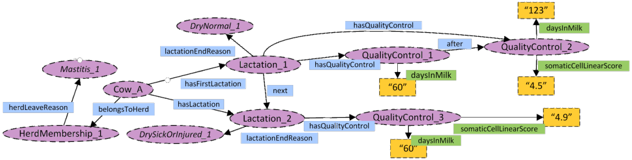


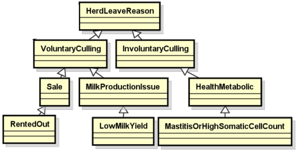
Dairy Cattle Performance Data





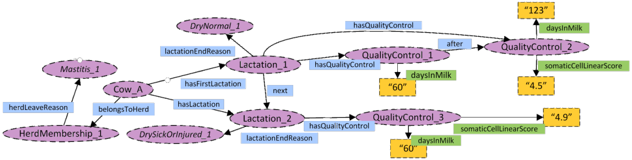
Dairy Cattle Performance Data

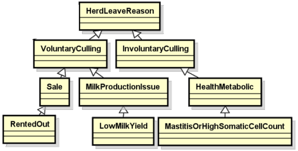




- For each cow, there is a (knowledge) graph recording events

Dairy Cattle Performance Data





- For each cow, there is a (knowledge) graph recording events
- Are there some reoccurring patterns?

Generalized Subgraphs

A pattern graph H is a *generalized subgraph* (with respect to an ontology O) of a graph G if

Generalized Subgraphs

A pattern graph H is a *generalized subgraph* (with respect to an ontology O) of a graph G if

- there is a subgraph isomorphism from H' to G

Generalized Subgraphs

A pattern graph H is a *generalized subgraph* (with respect to an ontology O) of a graph G if

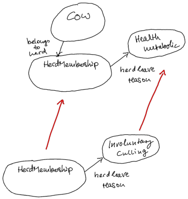
- there is a subgraph isomorphism from H' to G
- H' can be constructed by replacing vertex labels in H with *more specific labels* (according to our ontology)

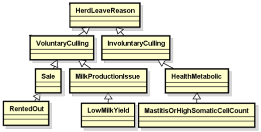
Generalized Subgraphs

Frequent Generalized Subgraph Mining via Graph Edit Distances

A pattern graph H is a *generalized subgraph* (with respect to an ontology O) of a graph G if

- there is a subgraph isomorphism from H' to G
- H' can be constructed by replacing vertex labels in H with *more specific labels* (according to our ontology)

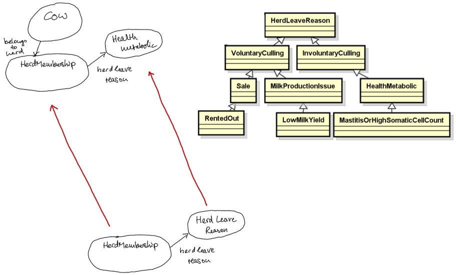



HerdLeaveReason
VoluntaryCulling
InvoluntaryCulling
Sale
MilkProductionIssue
HealthMetabolic
RentedOut
LowMilkYield
MastitisOrHighSomaticCellCount

Generalized Subgraphs

A pattern graph H is a *generalized subgraph* (with respect to an ontology O) of a graph G if

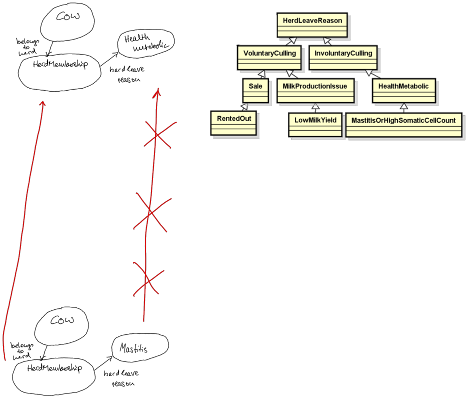
- there is a subgraph isomorphism from H' to G
- H' can be constructed by replacing vertex labels in H with *more specific labels* (according to our ontology)



Generalized Subgraphs

A pattern graph H is a *generalized subgraph* (with respect to an ontology O) of a graph G if

- there is a subgraph isomorphism from H' to G
- H' can be constructed by replacing vertex labels in H with *more specific labels* (according to our ontology)



Generalized Subgraph Mining

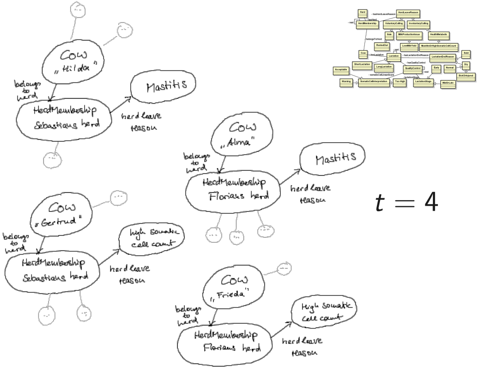
The *Frequent Generalized Subgraph Mining Problem* is then:

Generalized Subgraph Mining

Frequent Generalized Subgraph Mining via Graph Edit Distances

The *Frequent Generalized Subgraph Mining Problem* is then:

Given: A database D of graphs, an ontology O and a frequency threshold t



$t = 4$

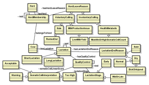
Generalized Subgraph Mining

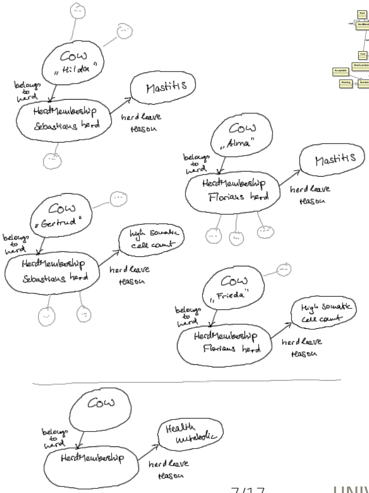
Frequent Generalized Subgraph Mining via Graph Edit Distances

The *Frequent Generalized Subgraph Mining Problem* is then:

Given: A database D of graphs, an ontology O and a frequency threshold t

Compute: The set of all graphs that are generalized subgraphs of at least t graphs in D

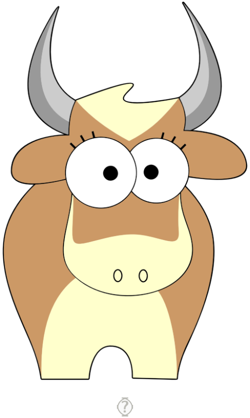




$t = 4$

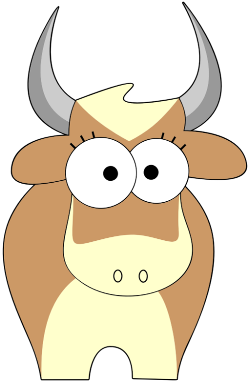
How Can This Help with Predictive Maintenance?

- Not automatically...



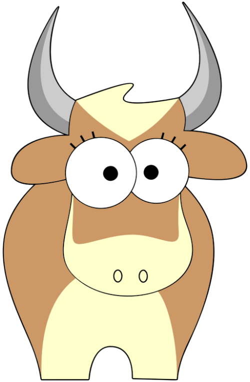
How Can This Help with Predictive Maintenance?

- Not automatically...
- But domain experts can interpret (smaller) frequent patterns



How Can This Help with Predictive Maintenance?

- Not automatically...
- But domain experts can interpret (smaller) frequent patterns
- And they can interpret changes in frequent patterns



Existing Approaches

Frequent Generalized Subgraph Mining has a long history

- ?
- ?

Existing Approaches

Frequent Generalized Subgraph Mining has a long history

- ?
- ?

And recently gained more traction

- ?
- ?
- ?

Existing Approaches

Frequent Generalized Subgraph Mining has a long history

- ?
- ?

And recently gained more traction

- ?
- ?
- ?

These papers all modify classical frequent subgraph mining algorithms.

Existing Approaches

Frequent Generalized Subgraph Mining has a long history

- ?
- ?

And recently gained more traction

- ?
- ?
- ?

These papers all modify classical frequent subgraph mining algorithms.

As a subroutine, they use subgraph isomorphism algorithms.

Our Approach

Proposal

We propose to replace subgraph isomorphism computations by *graph edit distance* computations

Proposal

We propose to replace subgraph isomorphism computations by *graph edit distance* computations

- This makes the problem *harder*

Proposal

We propose to replace subgraph isomorphism computations by *graph edit distance* computations

- This makes the problem *harder*
- But it allows some nice *freedom and simplicity* in modeling

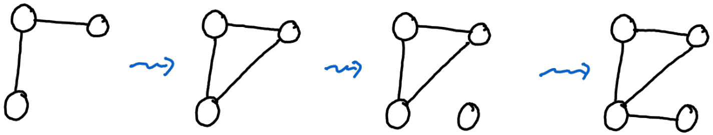
Proposal

We propose to replace subgraph isomorphism computations by *graph edit distance* computations

- This makes the problem *harder*
- But it allows some nice *freedom and simplicity* in modeling
- And it *simplifies* rather intricate mining algorithms

Graph Edit Distance

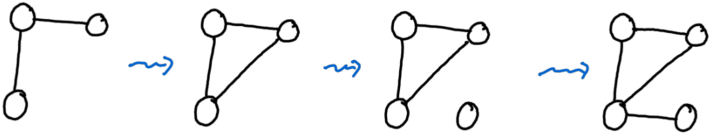
The *graph edit distance* between G and H is the smallest cost of a sequence of edits transforming G into H.



Graph Edit Distance

Edit operation	Edit cost
Insert an isolated vertex with label $\alpha \in \Sigma$	$c(\varepsilon, \alpha)$
Delete an isolated vertex u	$c(\lambda(u), \varepsilon)$
Substitute the label of a vertex u by $\alpha \in \Sigma$	$c(\lambda(u), \alpha)$
Insert an edge with label $\alpha \in \Sigma$	$c(\varepsilon, \alpha)$
Delete an edge e	$c(\lambda(e), \varepsilon)$
Substitute the label of an edge e by $\alpha \in \Sigma$	$c(\lambda(e), \alpha)$

The *graph edit distance* between G and H is the smallest cost of a sequence of edits transforming G into H.



A Nice Polynomial Time Reduction

The GED can be used to solve the *subgraph isomorphism problem (SGI)* by imposing the following three constraints on the edit cost function:

A Nice Polynomial Time Reduction

The GED can be used to solve the *subgraph isomorphism problem (SGI)* by imposing the following three constraints on the edit cost function:

$$\forall \beta \in \Sigma_\varepsilon \colon c(\varepsilon, \beta) = 0 \qquad \text{(free insertions)}$$

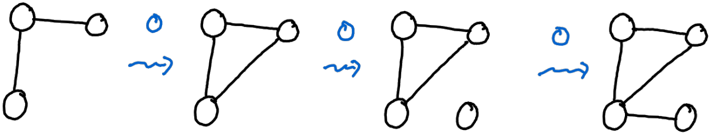
$$\forall \alpha \in \Sigma \colon c(\alpha, \varepsilon) > 0 \qquad \text{(paid deletions)}$$

$$\forall \alpha, \beta \in \Sigma \colon c(\alpha, \beta) > 0 \iff \alpha \neq \beta \qquad \text{(paid substitutions)}$$

A Nice Polynomial Time Reduction

The GED can be used to solve the *subgraph isomorphism problem (SGI)* by imposing the following three constraints on the edit cost function:

$$
\begin{aligned}
\forall \beta \in \Sigma_\varepsilon \colon c(\varepsilon, \beta) &= 0 && \text{(free insertions)} \\
\forall \alpha \in \Sigma \colon c(\alpha, \varepsilon) &> 0 && \text{(paid deletions)} \\
\forall \alpha, \beta \in \Sigma \colon c(\alpha, \beta) &> 0 \iff \alpha \neq \beta && \text{(paid substitutions)}
\end{aligned}
$$



A Nice Polynomial Time Reduction

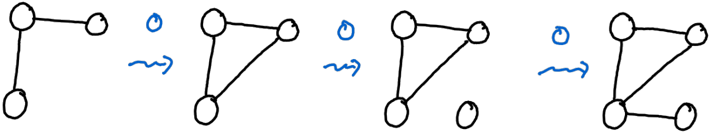
The GED can be used to solve the *subgraph isomorphism problem (SGI)* by imposing the following three constraints on the edit cost function:

$$\forall \beta \in \Sigma_\varepsilon \colon c(\varepsilon, \beta) = 0 \qquad \text{(free insertions)}$$

$$\forall \alpha \in \Sigma \colon c(\alpha, \varepsilon) > 0 \qquad \text{(paid deletions)}$$

$$\forall \alpha, \beta \in \Sigma \colon c(\alpha, \beta) > 0 \iff \alpha \neq \beta \qquad \text{(paid substitutions)}$$

$$\mathrm{SGI}(H, G) = \text{true} \iff \mathrm{GED}(H, G) = 0.$$



A Nice Polynomial Time Reduction

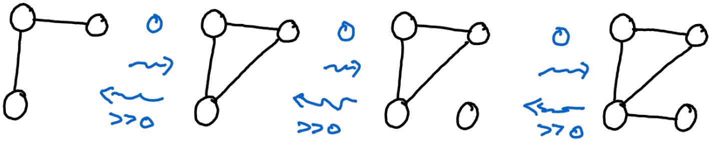
The GED can be used to solve the *subgraph isomorphism problem (SGI)* by imposing the following three constraints on the edit cost function:

$$\forall \beta \in \Sigma_\varepsilon \colon c(\varepsilon, \beta) = 0 \qquad \text{(free insertions)}$$

$$\forall \alpha \in \Sigma \colon c(\alpha, \varepsilon) > 0 \qquad \text{(paid deletions)}$$

$$\forall \alpha, \beta \in \Sigma \colon c(\alpha, \beta) > 0 \iff \alpha \neq \beta \qquad \text{(paid substitutions)}$$

$$\mathrm{SGI}(H, G) = \text{true} \iff \mathrm{GED}(H, G) = 0.$$



Another Nice Polynomial Reduction

To solve the *generalized subgraph isomorphism problem (GSGI)*, we impose the following four constraints on the edit cost function:

Another Nice Polynomial Reduction

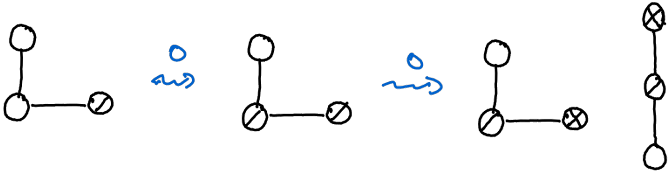
To solve the *generalized subgraph isomorphism problem (GSGI)*, we impose the following four constraints on the edit cost function:

$$
\begin{aligned}
\forall \beta \in \Sigma_\varepsilon &: c(\varepsilon, \beta) = 0 && \text{(free insertions)} \\
\forall \alpha \in \Sigma &: c(\alpha, \varepsilon) > 0 && \text{(paid deletions)} \\
\forall \alpha, \beta \in \Sigma &: c(\alpha, \beta) > 0 \iff \alpha \neq \beta \text{ and } \alpha \text{ is not more general than } \beta && \text{(paid substitutions)} \\
\forall \alpha, \beta \in \Sigma &: c(\alpha, \beta) = 0 \iff \alpha = \beta \text{ or } \alpha \text{ is more general than } \beta && \text{(free specializations)}
\end{aligned}
$$

Another Nice Polynomial Reduction

To solve the *generalized subgraph isomorphism problem (GSGI)*, we impose the following four constraints on the edit cost function:

$$\begin{aligned}
\forall \beta \in \Sigma_\varepsilon \colon c(\varepsilon, \beta) &= 0 && \text{(free insertions} \\
\forall \alpha \in \Sigma \colon c(\alpha, \varepsilon) &> 0 && \text{(paid deletions} \\
\forall \alpha, \beta \in \Sigma \colon c(\alpha, \beta) &> 0 \iff \alpha \neq \beta \text{ and } \alpha \text{ is not more general than } \beta && \text{(paid substitutions} \\
\forall \alpha, \beta \in \Sigma \colon c(\alpha, \beta) &= 0 \iff \alpha = \beta \text{ or } \alpha \text{ is more general than } \beta && \text{(free specializations}
\end{aligned}$$



Another Nice Polynomial Reduction

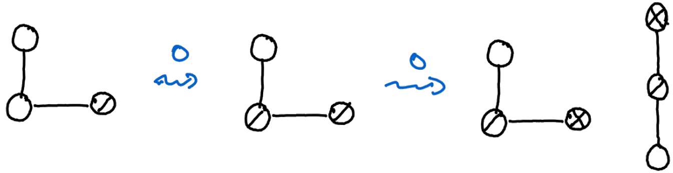
To solve the *generalized subgraph isomorphism problem (GSGI)*, we impose the following four constraints on the edit cost function:

$$\forall \beta \in \Sigma_\varepsilon \colon c(\varepsilon, \beta) = 0 \qquad \text{(free insertions}$$

$$\forall \alpha \in \Sigma \colon c(\alpha, \varepsilon) > 0 \qquad \text{(paid deletions}$$

$$\forall \alpha, \beta \in \Sigma \colon c(\alpha, \beta) > 0 \iff \alpha \neq \beta \text{ and } \alpha \text{ is not more general than } \beta \qquad \text{(paid substitutions}$$

$$\forall \alpha, \beta \in \Sigma \colon c(\alpha, \beta) = 0 \iff \alpha = \beta \text{ or } \alpha \text{ is more general than } \beta \qquad \text{(free specializations}$$



Another Nice Polynomial Reduction

To solve the *generalized subgraph isomorphism problem (GSGI)*, we impose the following four constraints on the edit cost function:

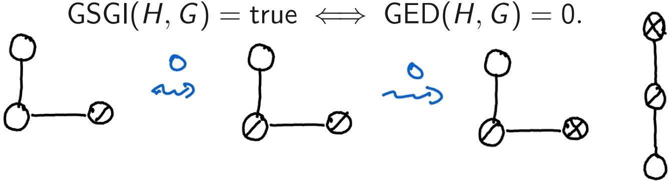
$$\forall \beta \in \Sigma_\varepsilon \colon c(\varepsilon, \beta) = 0 \qquad \text{(free insertions)}$$

$$\forall \alpha \in \Sigma \colon c(\alpha, \varepsilon) > 0 \qquad \text{(paid deletions)}$$

$$\forall \alpha, \beta \in \Sigma \colon c(\alpha, \beta) > 0 \iff \alpha \neq \beta \text{ and } \alpha \text{ is not more general than } \beta \qquad \text{(paid substitutions)}$$

$$\forall \alpha, \beta \in \Sigma \colon c(\alpha, \beta) = 0 \iff \alpha = \beta \text{ or } \alpha \text{ is more general than } \beta \qquad \text{(free specializations)}$$

$$\mathrm{GSGI}(H, G) = \text{true} \iff \mathrm{GED}(H, G) = 0.$$



+-

Frequent Generalized Subgraph Mining

- There are fast heuristics

Frequent Generalized Subgraph Mining

- There are fast heuristics
- There was some nice work on lower bounds at ECMLPKDD this year which might be adapted

Frequent Generalized Subgraph Mining

  - There are fast heuristics
  - There was some nice work on lower bounds at ECMLPKDD this year which might be adapted
- We say that a graph is a generalized subgraph if it has a small generalized subgraph edit distance

Implementation

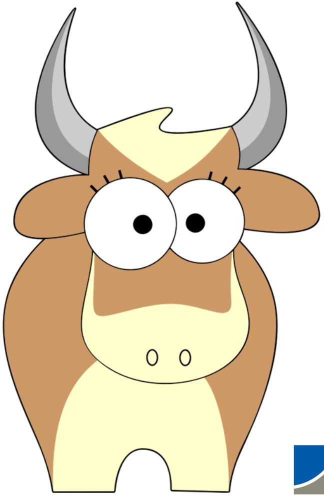
- We implemented a proof of concept graph mining algorithm

Implementation

- We implemented a proof of concept graph mining algorithm
- It is available on `https://github.com/RichardPalme/fasm`

Conclusion

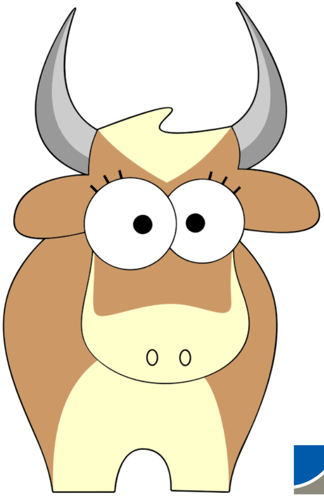
- Generalized Subgraphs can be identified with a specialized variant of the graph edit distance



Conclusion

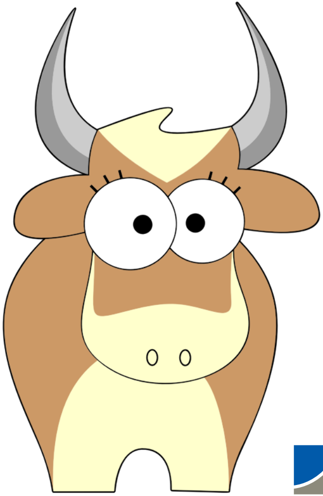
Frequent Generalized Subgraph Mining via Graph Edit Distances

- Generalized Subgraphs can be identified with a specialized variant of the graph edit distance
- This allows to mine generalized patterns in an elegant way



Conclusion

- Generalized Subgraphs can be identified with a specialized variant of the graph edit distance
- This allows to mine generalized patterns in an elegant way
- We can include interesting costs (checkout the paper) to make the mining practically better



References I