Querying and Creating Visualizations by Analogy

Carlos E. Scheidegger, Huy T. Vo, David Koop, Juliana Freire, Cláudio T. Silva

SCI Institute, School of Computing
University of Utah
Outline

• Provenance **reuse**
  • We have all this rich metadata - let’s use it

• Query-by-example
• Visualization by Analogy

• (VisTrails intro)
  • **Transparent provenance tracking**
Related Work

- Visualization Systems and Libraries
  - AVS, DX, SCIRun, VTK

- History tracking and formalisms
  - Jankun-Kelly et al’s pset-calculus
  - Kreuseler et al, VDM history
  - Brodlie’s et al’s GRASPARC
  - VisTrails
Provenance

- The “pedigree” of an artifact
- Where did it come from? Who held it?

From the Tour: The Beginnings of Impressionist Landscape
Object 5 of 7

Provenance


Provenance in VisTrails

- **Process** provenance
- How was this visualization created?
- Persistent
- Transparent

- Reuse
  - Can we do better than just presenting?
Why not query languages?

```sql
Select ExecutableWorkflowId, Execution_Event.ExecutionId, Event.EventId,
       Execution_Event.ExecutableWorkflow_ExecutableActivityId
from Execution, Execution_Event, Event, Event_Property_Value, Property, Value
where Value=Cast('C:\TEMP\atlas-x.gif' as binary) and
Event_Property_Value.PropertyId=Property.PropertyId and Event_Property_Value.
and Event.EventId=Event_Property_Value.EventId and Execution_Event.EventId
Execution_Event.ExecutionId=Execution.ExecutionId;
```
Why not query languages?

```sql
Select ExecutableWorkflowId, Execution_Event.ExecutionId, Event.EventId,
Execution_Event.ExecutableWorkflow_ExecuteableActivityId
from Execution, Execution_Event, Event, Event_Property_Value, Property, Value
where Value=Cast('C:\TEMP\atlas-x.gif' as binary) and
```

```
wf{*}: upstream(x) union x where
x.module = “SoftMean” and executed
(x) and y in upstream(x) and
y.module = “AlignWarp” and
y.parameter(“model”) = “12”
```
Why not query languages?

This is still only mildly better than straight SQL... Does not expose mapping to relational schema

```sql
```

```
wf{*}: upstream(x) union x where x.module = “SoftMean” and executed (x) and y in upstream(x) and y.module = “AlignWarp” and y.parameter(“model”) = “12”
```
Query-by-Example

- Do not teach the user new forms of interaction!
Visualization by Analogy

- Create new visualizations by saying “do as they did”
- Specify **what**, not **how**

www.vistrails.org
Query-by-Example

- Trivially reducible from MAX-CLIQUE
  - ... and MAX-CLIQUE is NP-Complete
  - ... and MAX-CLIQUE is fundamentally hard to approximate
- Solution: algorithm tailored to problem domain
Query-by-Example

- Split every subgraph in topologically sorted layers
- Ok, since all pipelines are DAGs in VisTrails
Query-by-Example

- Now search for layers that are connected in the same way in the database
Now search for layers that are connected in the same way in the database
Query-by-Example

- Now search for layers that are connected in the same way in the database

Query

1
2
3

Database

1
2
3
4
5

Match

No match
Query-by-Example

- Now search for layers that are connected in the same way in the database
Query-by-Example

• Might return false positives - it ignores the particular connectivity between topological layers

• Not too harmful - most modules cannot connect to one another
Query-by-Example

- Might return false positives - it ignores the particular connectivity between topological layers

- Not too harmful - most modules cannot connect to one another
- Might return false positives - it ignores the particular connectivity between topological layers

- Not too harmful - most modules cannot connect to one another
Query-by-Example

- Might return false positives - it ignores the particular connectivity between topological layers

- Not too harmful - most modules cannot connect to one another
• A version tree stores a set of **actions**
  • Each action is a function on the set of all possible visualizations: $\mathcal{V} \rightarrow \mathcal{V}$
  • $a_n \circ a_{n-1} \circ a_{n-2} \cdots \circ a_0$
• We can use those to determine the difference between visualizations
• Moving up, then down the version tree
• Action to go from A to B is $a_3 \circ a_2 \circ a_0^{-1} \circ a_1^{-1}$
• A diff is a template: reapply it elsewhere

• How do we match two pipelines?
Algorithm Overview

- Compute the difference
- Compute the map
- Apply $\text{map}_{ac}$ to $\delta_{ab}$
- Compute the new pipeline

\[
\delta_{ab} = \Delta(p_a, p_b)
\]

\[
\text{map}_{ac} = \text{map}(p_a, p_c)
\]

\[
\delta_{cb} = \text{map}_{ac}(\delta_{ab})
\]

\[
p_d = \delta_{cb}(p_c)
\]
• Simplest version is again reducible from MAX-CLIQUE
• We will now use a probabilistic argument to create a Markov chain
How does it work?

- Module compatibility: **prior**
  - $f : M^2 \rightarrow [0, 1]$
  - **Independent** of graph topology

- Probability of match between a pair
  - **Dependent** of graph topology
  - Linear combination of probability of match in the neighborhood pairs and data

- This is a Markov chain!
How does it work?

- Graph product $G$ of the two input graphs
  - each vertex in $G$ represents a possible match
  - similarity is then defined as

$$
\pi = \alpha A(G)\pi + (1 - \alpha)c(G)
$$

$$
= M_G \pi
$$

- $\pi$ is an eigenvector of $M_G$
  - It is the limit distribution of the transition matrix
How does it work?

\[ G_A \times G_B \]
How does it work?

$G_A \times G_B$
How does it work?

$G_A \times G_B$
How does it work?

$G_A 	imes G_B$
How does it work?

\[ G_A \times G_B \]

- \( G_A \)
- \( G_B \)
How does it work?

Each node is assigned some initial value. (It doesn’t matter which, as long as the values sum to one!)
How does it work?

$p_k(a_0 \rightarrow b_0)$

$p_k(a_0 \rightarrow b_1)$

$p_k(a_0 \rightarrow b_2)$

$p_k(a_0 \rightarrow b_3)$

$p_k(a_1 \rightarrow b_0)$
How does it work?

\[ p_{k+1}(a_0 \rightarrow b_0) = (1 - \alpha) c(a_0, b_0) + \frac{\alpha}{3} \]

\[ pk(a_0 \rightarrow b_0) \]

\[ pk(a_0 \rightarrow b_3) \]

\[ pk(a_0 \rightarrow b_2) \]

\[ pk(a_1 \rightarrow b_0) \]

\[ pk(a_0 \rightarrow b_1) \]

\[ pk(a_0 \rightarrow b_1) + \]

\[ pk(a_0 \rightarrow b_1) + \]

\[ pk(a_1 \rightarrow b_0) \]
How does it work?

\[ p_{k+1}(a_0 \rightarrow b_0) = (1 - \alpha)c(a_0, b_0) + \frac{\alpha}{3} \left( p_k(a_0 \rightarrow b_3) + p_k(a_0 \rightarrow b_1) + p_k(a_1 \rightarrow b_0) \right) \]
$p_{k+1}(a_0 \rightarrow b_0) = (1 - \alpha)c(a_0, b_0) + \alpha/3 (p_k(a_0 \rightarrow b_3) + p_k(a_0 \rightarrow b_1) + p_k(a_1 \rightarrow b_0))$
How does it work?

$$p_{k+1}(a_0 \rightarrow b_0) = (1 - \alpha)c(a_0, b_0) + \alpha/3 \left( p_k(a_0 \rightarrow b_3) + p_k(a_0 \rightarrow b_1) + p_k(a_1 \rightarrow b_0) \right)$$

Do it for all nodes, until convergence
How does it work?

- $\pi$ is defined over graph product
- For each module in the second pipeline, pick maximal value of $\pi$ on first pipeline: this is the match
- Many others possible
The matching algorithm
The matching algorithm
The matching algorithm
The matching algorithm
• Analogies are not fool-proof
• Creating a complex visualization out of simple ones

• (demo)
Discussion

- If your system can encode actions as functions on the space of objects of interest, store these explicitly.
- That will be your “version tree” - everything else is just the same.
- Easy to incorporate domain-specific knowledge in analogies: change $A(G)$ and $c(G)$. 
Acknowledgments

• Sarang Joshi, Suresh Venkatasubramanian, Erik Anderson, João Comba
• VisTrails dev team
• Many open source packages and devs: VTK, SciPy, teem, matplotlib
• VisTrails is open source! http://www.vistrails.org
  • Shameless plug: Visit the SCI booth!
• NSF, DOE, IBM Faculty Award
Thank you!

- Questions?
Too much data

- We are better off with visualization systems than without - but it’s still pretty messy
Creating and Querying Visualizations by Analogy

SCI Institute, School of Computing

UNIVERSITY OF UTAH