Graph Querying Meets HCI: State of the Art and Future Directions

Sourav S Bhowmick
Nanyang Technological Univ
Singapore

Byron Choi
Hong Kong Baptist Univ
Hong Kong

Chengkai Li
The Univ of Texas at Arlington
USA
First Generation Data Management

Driven primarily by enterprises to store and query data

- **Developers**: Build DB & applications
- **Business analysts**: Pose queries
- **DB admin**: Tune & monitor performance
- **End users**: Generate data, query data

Emergence of Network Data

Social network

Ecological network

WWW

Human PPI network

The emergence of network maps:

- Movie Actor Network, 1998;
- World Wide Web, 1999;
- C elegans neural wiring diagram 1990
- Citation Network, 1998
- Metabolic Network, 2000;
- PPI network, 2001
Querying Graphs

Query Formulation

• Formal query language
  • SPARQL, Cypher

Query Processing

• Efficient algorithms and optimization techniques to process queries “quickly”

A large set of small/medium-sized graphs
A large graph/network
Massive graph
Fifth Generation Data Management

Data management has democratized

### End Users:
Generator, processor, and consumer

- DB illiterate
- Increasingly complex data and computation
- Resides everywhere

Querying Graphs: The First Generation Approach

```
1 prefix wp: <http://vocabularies.wikipathways.org/wp#>
2 prefix dcterms: <http://purl.org/dc/terms/>
3 prefix foaf: <http://xmlns.com/foaf/0.1/>
4
5 select (str(?organismName) as ?organism) ?page ?gene1 ?gene2 ?interaction where {
6   ?gene1 a wp:GeneProduct .
7   ?gene2 a wp:GeneProduct .
8   ?interaction wp:source ?gene1 ;
9     wp:target ?gene2 ;
10    a wp:Conversion ;
13   wp:organismName ?organismName .
14   FILTER (?gene1 != ?gene2)
15 } ORDER BY ASC(?organism)
```
Thirty years of research on query languages can be summarized by: we have moved from SQL to XQuery. At best we have moved from one declarative language to a second declarative language with roughly the same level of expressiveness. It has been well documented that end users will not learn SQL; rather SQL is notation for professional programmers.

The Lowell Database Research Self-Assessment, Communication of the ACM (May 2005)
Usability Matters!

**Usability**

**Data-driven Software**

- **Performance**
- **Functionality**
Usability [Preece et al.]

What is it?
How well users can use the system’s functionality

Dimensionality

- **Learnability**: is it easy to learn?
- **Efficiency**: once learned, is it fast to use?
- **Memorability**: is it easy to remember what you learned?
- **Errors**: are errors few and recoverable?
- **Satisfaction**: is it enjoyable to use?
Visual Graph Querying

Usability and good UI design are closely related
Different Worlds

DB World

- Functionalities
- Performance
- Usability

HCI World

- Functionalities
- Performance
- Usability
The Chasm for 40+ Years
Graph Querying Meets HCI
Lessons from HCI: Schneiderman’s 8 Golden Rules

- Strive for consistency.
- Give shortcuts to the user.
- Offer informative feedback.
- Make each interaction with the user yield a result.
- Offer simple error handling.
- Permit easy undo of actions.
- Let the user be in control.
- Reduce short-term memory load on the user.
Tutorial Overview

- Visual Interfaces for Graph Queries
- Visual Graph Query Formulation & Guidance
- Visual Action-aware Graph Query Processing
- Query Results Exploration & Visualization

- Strive for consistency
- Make each interaction with the user yield a result
- Let the user be in control
- Give shortcut to the user
- Informative feedback
- Simple error handling
Visual Interfaces for Graph Queries
Visual Graph Query Interfaces

- Manual
- Data-driven

Functionalities vs Aesthetics
Manual Visual Graph Query Interfaces
Manual Visual Graph Query Interfaces
Hardcoded labels, patterns
Limited variety
Manual maintenance
Not portable
Data-driven Visual Interface Construction & Maintenance

Graph Repository

Auto

Diverse content

Portable

Auto maintenance
Data-driven Construction

Content selection

• Which patterns should be in the palette?
  • Formulate query easily and faster
  • Give shortcuts

• Issues
  • Size of the palette
  • Maximally covers the DB
  • Minimal redundancy among patterns
  • Aesthetics-aware
Content Maintenance

• How do we maintain the labels and patterns as underlying data changes?

• Issues
  • Real-time maintenance
  • Batch vs Incremental
  • Enhance usability (gain in coverage and reduction in redundancy)
  • Leverage usage patterns and query workload (if available)
DAVINCI: Initial Effort

[ICDE 15, VLDB 16]

Online

Canned patterns

Closure graphs

Offline

GraphDB

Large set of small/medium sized graphs

Graph set clustering

Topologically-similar partitions

Closure graph set generation
Visual Graph Query Formulation & Guidance
### Opportune Query Feedback

#### Modeling feedback
- An alert or notification for a secondary task when a user is working on a primary task

#### Needs
- Detect efficiently
- Notify **opportune**ly
  - Ineffective to notify at the end of query formulation

---

Delivering notifications inopportune can negatively impact task completion time, lead to more errors, and increase user frustration.
When to notify?

**Breakpoint**

- The moment of transition between two observable, meaningful units of task execution, and reflects a change in perception or action
  - [Newton, 1973]
- Coarse, Medium, and Fine
- Best moment to interrupt a user is on breakpoints between tasks
  - [Iqbal & Bailey, CHI 2008]
  - Defer the notification to appear in the next breakpoint

Adopt defer-to-breakpoint-based strategy for interrupting query formulation tasks

- Reduction of Interruption cost and frustration
- React faster to notifications
- Task-relevant notifications should be delivered at Medium or Fine breakpoints
Deliver notification before the construction of the succeeding query condition is finished (Optimal breakpoints)

How do we estimate the time available for delivery of notification at optimal breakpoint?
HCl-Inspired Quantitative Model

\[
T_m = a + b \log_2 \left( \sqrt{\frac{D^2}{W}} + \eta \frac{D^2}{H} + 1 \right)
\]

\[
T_s = m + n \times (\log_2(p + 1))
\]

\[
0 < T_{opt} < T_m + T_s
\]

[Accott & Zhai, CHI 03]

[Ahlstrom, CHI 05]
The iSERF Framework

[CIKM 15]

1. **Interruption-Sensitive Notification Module**

   - Cursor moving towards Schema Panel
     - Compute movement time $T_m$
     - Suspend notification by $T_m$ time

   - Cursor in Schema Panel
     - Compute selection time $T_s$
     - Suspend notification by $T_s$ time for item to be selected

   - Notification delivery
     - Deliver appropriate notification identifying condition(s) for empty result

2. **Empty Result Detection Module**
More on the Feedback Module

Query Autocompletion

Action Guidance
Query Autocompletion Demo

http://www.comp.hkbu.edu.hk/~csppyi/autog/
## Autocompletion Comparisons

<table>
<thead>
<tr>
<th>User Action</th>
<th>Keyword Search</th>
<th>Visual Graph Query</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User Action</strong></td>
<td>keystroke</td>
<td>Drag, click</td>
</tr>
<tr>
<td><strong>Atomic Unit</strong></td>
<td>char: 'a', 'b', 'c', ...</td>
<td>edge: C-C, C=C, ...</td>
</tr>
<tr>
<td><strong>Logical Unit</strong></td>
<td>keyword: &quot;world&quot;, &quot;clock&quot;, ...</td>
<td>subgraphs</td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>concatenated keywords</td>
<td>graphs</td>
</tr>
<tr>
<td></td>
<td>&quot;world clock&quot;, &quot;world cup&quot;, ...</td>
<td>C=C-C=C-C=C-C, ...</td>
</tr>
</tbody>
</table>
The AutoG Framework

[VLDB 16, VLDBJ 17]
User Preference / Intent

User intent value of a query (suggestion) set

\[
\text{util}(Q') = \alpha \times \frac{1}{k} \sum_{q' \in Q'} \text{sel}(q') + \beta \times \frac{1}{k(k - 1)} \sum_{q'_i, q'_j \in Q', i \neq j} \text{dist}(q'_i, q'_j)
\]

(MCCS) Distance between two graph suggestions

\[
\text{dist}(g_1, g_2) = 1 - \frac{|\text{cs}(g_1, g_2)|}{\max\{|g_1|, |g_2|\}}
\]

Property: \(\text{util}\) is submodular \(\rightarrow\) greedy
Optimizations

The FDAG index

- Index c-Prime features and their pairwise compositions
- Prune automorphic suggestions (redundant suggestions) early

Online ranking

- Approximate selectivities of query suggestions
- Prune empty suggestions early
- Optimize diversity computation
  - trimming the common parts between suggestions
More on the Feedback Module

Query Autocompletion

Action Guidance
Orion

Overview

- Interactive visual query builder with suggestions
- Iteratively suggest edges based on their relevance to the user’s query intent, according to the partial query graph so far
  - Edge ranking: query-specific random decision paths
- The use of statistics based on data graph, query logs, and so on.

Suggestions: Grey nodes/edges

- Accepted by users: Positive edges (become blue)
- Reject or ignored by users: Negative edges

User’s intent can be derived from these edges
Orion GUI

Dynamic list of all possible user actions at any given moment

Control panel for various settings and tips
Orion Implementation

- Prototype
  http://idir.uta.edu/orion

- Video Introduction
Visual Action-aware Graph Query Processing
Subgraph Queries

Subgraph Containment
- Given a graph DB $D$ and a query graph $Q$, find all data graphs in $D$ in which $Q$ is a subgraph
- Subgraph isomorphism from $Q$ to $G \in D$

Subgraph Similarity
- Data graphs that “approximately” contain the query graph
- Use subgraph distance based on maximum connected common subgraph (MCCS)
Classical Visual Querying Paradigm

40+ years old query paradigm!

GraphDB

Large collection of small graphs

Query formulation → Query processing

time

HCI → DB

40+ years old query paradigm!
Rethink the classical query paradigm

• Why wait for the complete visual query to be constructed before initiating query evaluation? How can we blend these two steps?
• By initiating query processing “early”, can we significantly reduce the system response time?
Non-traditional Challenges

- Partial query-aware indexing schemes
- Materialization of intermediate results
- Selectivity-free query processing
- Focus on waiting time of users

“Computing time (power) is getting cheaper but users’ time isn’t..”
Overview of VOGUE [SIGMOD 10, ICDE 12, CIDR 13]

A fragment g is frequent if its support is no less than $\alpha|D|$, or all subgraphs are frequent.

**Online**
- Candidate graph ids
- SPIG

**Offline**
- GraphDB
- Frequency subgraph Miner (gSpan)
- SPIG
- Candidate graph ids

- Subgraph isomorphism test (extension of VF2)
- DF-Index
- MF-Index
- A2I Index (DIFs)
QUBLE: Extension to Large Graphs [VLDB J 14]

Online

Offline

Graph Partitioning (METIS)

Graphlets
( Partition graphs, bridges )

|g|=1 or |g| = 2 and is an maximal cover graph (SIF)

A2F-Index

A2I Index

Supergraphlet-at-a-time

Frequent fragments

G-SPIG
Performance Summary

Outperforms traditional approaches in terms of waiting time

Not significantly impacted by query formulation sequence

Works well with small-sized queries

**LASER**: Newer version can handle large query graphs and scales to more than million data graphs (10X more than state-of-the-art)!
Challenges for Performance Study

Large-scale performance study

- Traditional approach
  - Randomly extract subgraphs of different size and execute them
- Doesn’t work in this paradigm!

Why?

- Queries need to be visually constructed by users
- GUI latency is critical for performance study

Challenge

- Users are expensive!
- How do we simulate visual query formulation?
VISUAL [ICDE 15, TKDE 17]

C1: Test Subgraph Query Generator

C2: Quantitative Model of Query Formulation

C3: Visual Subgraph Query Simulator

Index

Graph Repository

Data graphs

Test queries

Model
Quantitative Model for Query Formulation Time

\[ T(m) + T(s) + T(d) + T(e) \]

[Accott & Zhai, CHI 03]

[Cockburn et al, HCI 09]
VISUAL Demo
Query Results
Exploration & Visualization
Query Results Exploration

Two Categories

- Very few efforts!
- Large set of small graphs vs large networks

Large set of small graphs

- Typically a decision problem
- Highlight a subgraph that matches the query
- [SIGMOD 10, ICDE 12]
Query Results Exploration

Large Networks

- Summarization-based (SLQ \[SIGMOD 14\])
- Supergraphlet-at-a-time (QUBLE \[VLDBJ 14, SIGMOD 13\])
- Feature-based (R2DB \[ICDE 12\])
Conclusions
Bridging Usability, Performance, Functionality

Visual query interface

Query response time

Query feedback, Exploratory search

Usability

Software

HumanDB

Performance

Functionality
Shifting Traditions

1990-2015: Visual query interfaces are constructed manually

2015: Automatic, data-driven construction of visual graph query interface

1970s-2005s: Query Formulation → Query Processing

2006s: Visual query form. Query Processing

1990s - 2015: Visual query performances are carried out manually

2015: Automated query construction and performance benchmarking
Open Research Problems

- More complex graph queries: Homomorphism-based queries, multi-attribute queries, graph simulation
- Visually querying massive graphs
- How can we extend data-driven GUI construction to be aesthetics-aware?
- Multi-faceted exploration and visualization of query results
- HCl-awareness with other types of data?
Final Words

HCI-aware Data Management

- Towards usable data management systems
- Making visual query interface design data-driven
- Making query formulation & processing HCI-driven
- Novel area of research

Multi-disciplinary effort:

Data management

HCI

Cognitive psychology

Broad goal

Stimulating a cultural shift in our thinking by HCI, cognitive psychology and data management to “work” together
Thank You!