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

Sourav S Bhowmick
Nanyang Technological University
Singapore
assourav@ntu.edu.sg

Byron Choi
Hong Kong Baptist University
Hong Kong
bchoi@comp.hkbu.edu.hk

Chengkai Li
The University of Texas at Arlington
USA
cli@uta.edu

ABSTRACT

Querying graph databases has emerged as an important research problem for real-world applications that center on large graph data. Given the syntactic complexity of graph query languages (*e.g.,* SPARQL, Cypher), visual graph query interfaces make it easy for non-expert users to query such graph data repositories. In this tutorial, we survey recent developments in the emerging area of visual graph querying paradigm that bridges traditional graph querying with human computer interaction (HCI). We discuss manual and data-driven visual graph query interfaces, various strategies and guidance for constructing graph queries visually, interleaving processing of graph queries and visual actions, and visual exploration of graph query results. In addition, the tutorial suggests open problems and new research directions. In summary, in this tutorial we review and summarize the research thus far into HCI and graph querying in the database community, giving researchers a snapshot of the current state of the art in this topic, and future research directions.

1. INTRODUCTION

Graphs are a natural way of modeling data in a wide variety of domains and have been extensively studied in mathematics and many areas of computer science. Graph data in real-world applications such as biological and chemical databases (*e.g., PubChem*), social networks (*e.g., Twitter*), co-purchase networks (*e.g.,* Amazon.com), and information networks (*e.g., DBpedia*) has lead to a rejuvenation of research on graph data management and analytics. Several novel graph data management platforms have emerged from academia, industrial research labs (*e.g., Trinity*), and startup companies (*e.g., GraphX*). Several database query languages have been proposed for textually querying graph databases, *e.g.,* SPARQL, Cypher[1], and GraphQL [18].

Unfortunately, composing graph queries using aforementioned graph query languages often demands considerable cognitive effort from users and requires "programming" skill that is at least comparable to SQL [1]. A user must be familiar with the syntax of the language, and must be able to express her search goal accurately in a syntactically correct form. However, in many real life domains (*e.g.,* life sciences, social science, chemical science) it is unrealistic to assume that end users are proficient in such query languages. Hence, it is paramount to devise easy and intuitive techniques that reduce the burden of query formulation and thus increase the usability of graph databases.

Fortunately, unlike SQL, graph queries are more intuitive to draw than to compose them in textual format. Consequently, visual query interfaces (a.k.a GUI) that can enable an end user to draw a graph query interactively have gained increasing attention in recent times from academia [9, 16, 25, 26, 45] and industry (*e.g., PubChem*[2], *eMolecule*[3]). This has recently paved the way for research in a variety of directions that are driven by visual query interfaces such as novel visual query processing paradigm [25, 26], interactive guidance to users by providing relevant and timely feedback and suggestions during query formulation [23, 24, 39], exploration and visualization of graph query results [19, 45], among others.

This tutorial gives a comprehensive introduction to the topic of visual graph querying, discussing state of the art in the industry and in the academic world. In particular, a hallmark of this tutorial is to emphasize research efforts that aim to bridge two traditionally disparate topics in computer science, namely, graph querying and human computer interaction (HCI). Specifically, a key component we cover is the review of techniques and strategies that make visual graph query interface design *data-driven* and query formulation, query processing and results exploration HCI-*driven*. A brief overview of the scope of the tutorial is as follows.

- **Visual interfaces for graph querying:** We review different features and functionalities of a variety of visual graph query interfaces that are proposed by research communities and industry. We also review a recent effort in making such interface construction and maintenance *data-driven*.
- **Visual query formulation and guidance:** We cover various strategies for formulating different types of graph queries visually. Since a user-friendly visual query system is expected to interactively guide users into query construction, we review recent work in this direction in the context of visual query feedback (*e.g.,* empty result) and suggestions. A key emphasis here is work that leverage principles of HCI to guide users during query formulation.
- **Visual action-aware graph query processing:** In this part, we review HCI-*driven* graph query processing techniques. Specifically, in contrast to traditional query processing

[1] http://neo4j.com/docs/stable/cypher-query-lang.html

[2] http://pubchem.ncbi.nlm.nih.gov/

[3] https://www.emolecules.com/

SIGMOD'17, May 14-19, 2017, Chicago, IL, USA
© 2017 ACM. ISBN 978-1-4503-4197-4/17/05...$15.00
DOI: http://dx.doi.org/10.1145/3035918.3054774

Table 1: Tutorial overview.

Topic	*Representative papers*	*Demo*
Introduction		No
Visual interfaces for graph querying	[9, 12, 16, 25, 26, 39, 45], *Pubchem*, *eMolecule*	Yes ([26], *Pubchem)*
Visual graph query formulation and guidance	[5, 23, 24, 26, 39, 44, 47]	Yes ([23, 47])
Visual action-aware graph query processing	[4, 19, 25, 26]	Yes ([4, 26])
Query results exploration and visualization	[19, 26, 33, 43]	Yes ([19])
Future research direction		No

paradigm where a query is processed only after it is completely formulated, here we review efforts that interleave query processing with visual actions taken by users during query construction.

- **Query results exploration and visualization:** We review techniques proposed in the literature to explore and visualize results matches of a visual graph query. Specifically, we highlight efforts that make such exploration HCI-driven.
- **Future research directions:** : Finally, we discuss open problems on the topic of visual graph querying and providing possible directions for future work.

2. TUTORIAL OUTLINE

Our presentation follows a top-down approach, starting from visual query interface design, proceeding to visual query formulation and processing, visualization of query results, and concluding with future research directions in this arena. Table 1 shows the key papers discussed in this tutorial. Additionally, a key feature of our tutorial is live demonstration of various research prototypes related to the aforementioned topics as highlighted in the rightmost column in Table 1.

2.1 Visual Interfaces for Graph Querying

There have been considerable efforts on designing visual graph query interfaces (*i.e.,* GUI) for querying graph-structured data in a variety of application domains [9, 12, 16, 25, 26, 39, 45]. Interestingly, these GUIs provide diverse functionalities for formulating graph queries of varying complexity. For instance, some GUIs [7] provide a panel containing *canned patterns* to expedite query formulation whereas others do not [26, 39]. Intuitively, a *canned pattern* is a small topological pattern (*e.g.,* a benzene ring, triangle) which users can drag-and-drop into a query. Some may support approximate subgraph search but others may not. Additionally, some GUIs may suffer from poor aesthetics compared to others despite the fact that people prefer attractive interfaces and the effect of aesthetics in GUI appreciation is significant [11]. Hence, we begin our tutorial by summarizing various GUIs proposed in the literature and in industry for querying graph-structured data highlighting their distinguishing features, similarities and differences w.r.t to structure, functionalities, and aesthetics. A key point of our discussion is the emphasis on the fact that despite decades of research by the HCI community related to various theoretical models of visual tasks, menu design, and human factors, many of the current GUIs for graph querying are oblivious to these results.

Next, we focus on the construction and maintenance of visual query interfaces. There are currently two flavors for constructing and maintaining a GUI, namely, *manual* and *data-driven*. Most of the real-world GUIs follow the former approach (*i.e.,* they are constructed and maintained *manually*). That is, details of visual design of a GUI are manually worked out and contents of various components are created manually by "hard coding" them during GUI implementation. Unfortunately, such manual effort may create GUIs that do not provide sufficient features (*e.g.,* canned patterns) to aid query formulation, are static in nature when the underlying graph data repository evolves, and have limited portability [7]. To alleviate these limitations, we introduce a recent work on *data-driven* GUI construction and maintenance [48], which can automatically construct the content of various panels of a GUI and maintain them as underlying data evolves. Such a data-driven paradigm has several benefits such as superior support for visual subgraph query construction, significant reduction in the manual cost of maintaining an interface for any graph query-based application, and portability of the interface across diverse variety of graph querying applications.

2.2 Visual Graph Query Formulation and Guidance

Next, we review recent studies on techniques to facilitate visual graph query construction.

Visual graph query formulation. To formulate a graph query, there are mainly three approaches in the literature. In the *edge-at-a-time* approach (*e.g.,* [26]), a query is incrementally constructed by adding one edge at a time. In contrast, in the *pattern-at-a-time* approach (*e.g.,* https://pubchem.ncbi.nlm.nih.gov/edit2/index.html?cnt=0), one may compose a visual query by dragging and dropping canned patterns or subgraphs that are available on the GUI in addition to single edge construction. Observe that the latter approach is more efficient than the former as it typically takes lesser time to construct a query. More recently, VISAGE [39] realizes *Query By Example* (QBE) for formulating graph queries using examples.

Categories of query guidance. Regardless of the approach taken to construct a visual query, a user-friendly visual query system is expected to interactively guide users into constructing correct queries. A non-exhaustive list of the kinds of guidance such a system may employ is given below.

- *Syntactic and semantic help.* A visual query system can detect and notify users when there is a syntax or semantic problem in a query (*e.g.,* a missing quote on a string or a wrong type passed to a function). A system could even suggest better alternative syntax. These techniques can greatly enhance users' ability to formulate syntactically correct queries visually without resorting to memorizing various syntactic flavors of a query language.
- *Empty result feedback.* It is beneficial to end users if we can detect during query construction scenarios where a partially constructed query returns empty results and alert them in a timely fashion so that one can undertake appropriate remedial action(s). Several recent studies [5,39] address this problem by notifying a user opportunely when a partially constructed visual query yields an empty result.

- *Long-running query feedback.* A recent study [44] shows that a graph query that runs fast may slow down significantly after a slight modification to the query's structure. Unexpected query behaviors like this can confuse and annoy users of visual query systems, especially in the context of exploratory search. It would be better if the system could warn a user about a potentially long-running query fragment during query formulation. The study in [44] demonstrates a system in this direction.
- *Query fragments suggestion.* Given a partially-constructed visual graph query, it is useful to suggest top-*k* possible query fragments that the user may potentially add to her query in the subsequent step. Such suggestions can enhance user experience by greatly reducing the query formulation time. The recent query log-driven effort in [23], which suggests *edge increments* to the current query graph, is a step in this direction. However, since it provides only edge suggestions, the query formulation process may take many steps and users can only choose limited structural information. In contrast, AUTOG [47] suggests *subgraph increments*, enabling formulation of query graphs quickly. VISAGE [39] supports suggestions on nodes, edges, as well as conditions.

There are two common threads in the above scenarios. First, without the aid of a visual query feedback mechanism, users would be unaware of these cases, since each case requires comprehensive knowledge of the underlying data or query language. Feedback about various problems encountered during query construction as well as possible solutions is critical to enhancing the usability of a visual querying system. Second, as query conditions in a visual querying environment are typically constructed iteratively, it is often critical to detect and notify the aforementioned issues *opportunely*. It is ineffective to provide feedback at the *end* of query formulation. For instance, consider the empty result problem. It is ineffective to provide feedback *after* the query formulation as a user may have wasted her time and effort in formulating additional query conditions. Similarly, it is ineffective if the visual querying scheme fails to alert the user opportunely when a "long-running" query fragment is detected. Such opportune notification is also important in the context of query fragment suggestion as it is not beneficial if suggestions are made after the query has been visually constructed.

Interruption due to query feedback. The aforementioned notifications, however, come with a cost: they *interrupt* a primary task (*i.e.,* query formulation). This is because notifications divert attention [21, 38]. Intuitively, an *interruption* is a distraction that causes one to stop a scheduled task to respond to a stimulus. Many studies in the cognitive psychology and HCI communities have reported that interrupting users engaged in tasks by delivering notifications inopportunely can negatively impact task completion time, lead to more errors, and increase user frustration [2, 3, 21, 22, 37]. For instance, suppose a user is notified intrusively (*e.g.,* invoking a pop-up dialog box, highlighting a condition) of an empty result (due to previously formulated condition) when she is constructing a new edge in a query graph. This interruption may frustrate her as mental resources allocated for the current task are disrupted. Such inopportune, intrusive feedback adversely affects the usability of the system.

There exists little work that systematically explores the issue of delivering feedback opportunely by ensuring such notification is "interruption-sensitive". In [5], an interruption-sensitive framework is proposed in the context of the empty results problem. Specifically, it bridges the classical visual query feedback problem with intelligent notification management from the domains of HCI and cognitive psychology to notify the user about the following:

- Occurrence of the empty results problem due to the constructed query fragment *opportunely* so that she can avoid wasting time and effort in continuing constructing new conditions.
- Identify the constructed query condition(s) that is responsible for an empty result.
- Optionally, suggest a query modification to mitigate the empty results problem.

2.3 Visual Action-aware Graph Query Processing

Given a graph query formulated using a visual query interface, the next issue is the efficient processing of the query to retrieve matching results. There is a large body of research in the literature for efficient and scalable processing of a variety of graph queries on a set of small/medium sized graphs and on a large network [17, 28]. We can classify these efforts into two categories, namely, *visual action-unaware* and *visual action-aware*. In *visual action-unaware* graph query processing, the query processor remains idle when the query is being constructed. It only starts processing the query once the `Run` icon is clicked after the complete query is composed by the user. Hence, any traditional graph query processing technique can be utilized to process such a visual query. On the other hand, in *visual action-aware* graph query processing, instead of processing a query graph after its construction, visual query construction and processing is *interleaved* [19, 25, 26]. That is, the query processor utilizes the latency offered by the GUI actions to evaluate a partially constructed query graph at each step during query construction to generate candidate matches.

In this tutorial, we focus on visual action-aware graph query processing instead of its visual action-unaware counterpart as the techniques to process latter have been discussed in several tutorials in major data management venues [31]. We group the review of visual action-aware query processing techniques into two categories.

Query processing on a set of small data graphs. Recent work [25, 26] have proposed novel *visual action-aware feature-based indexes* and *query processing techniques* to support exact and approximate subgraph queries efficiently on a large collection of small or medium-sized data graphs (*e.g.,* chemical compounds). Specifically, when a user draws a new edge or a canned pattern on the query canvas, candidate data graphs containing the current query fragment are efficiently retrieved and monitored by leveraging the indexes. This process is repeated until the user clicks on the `Run` icon to signify the end of the query formulation step. Consequently, the final query results are generated from the prefetched candidate data graphs by performing verification test whenever necessary. Note that as this step invokes the subroutine for subgraph isomorphism test, the verification process needs to be minimized by judiciously filtering as many false candidates as possible without any verification test. These frameworks also support efficient maintenance of candidate data graphs in the presence of query modification during formulation.

Query processing on a large network. It is well-known that techniques for processing queries on a large set of small or medium-sized graphs are different from those employed for processing large or massive networks [17]. Here we review a recent research [19] on realizing the aforementioned query processing paradigm on a large network (*e.g.,* protein interaction networks and social networks). First, it decomposes a large network into a set of *graphlets* and *supergraphlets* using a minimum cut-based graph partitioning technique. Next, it mines a variety of frequent and infrequent fragments from them and identifies their occurrences in these graphlets and supergraphlets. Then, the indexing framework of [26] is augmented so that the mined fragments can be exploited to index graphlets for efficient blending of visual subgraph query formulation and query processing.

Automated performance study. User studies are the *sine qua non* for evaluating the performance and effectiveness of the aforementioned visual action-aware query processing techniques. For example, consider the visual subgraph querying paradigm in [25,26]. In contrast to the traditional query processing paradigm where the runtime performance of a large number of subgraph queries can be easily measured by automatically extracting a random collection of subgraphs from the underlying data and executing them [28], each visual query graph *must* be formulated by a set of users. This is because in this paradigm the availability of the GUI latency at each formulation step is exploited to prefetch and refine candidate matches. To address this challenge, we review a novel *synthetic visual subgraph query simulator* [4], which focuses on simulating subgraph query construction on a database containing a large number of small or medium-sized graphs. Using this framework, one can automatically generate many test subgraph queries having different *user-specified characteristics* (*e.g.,* frequent, infrequent) using indexes and simulate their formulation based on different query formulation sequences *without requiring human users*. Specifically, it employs an HCI-inspired *quantitative model* to estimate the time for subgraph query formulation and an algorithm to simulate visual query formulation of the generated test queries by leveraging the quantitative model. This framework can then be used to automate exhaustive evaluation of performances of various visual action-aware query processing techniques.

2.4 Query Results Exploration and Visualization

Although there have been several work in the literature on interactive data exploration [20] and visualization of graph analysis [27,32,42], very few have focused on rich and interactive exploration and visualization of graph query results. We can categorize exploration and visualization of graph query results into two types. For a large collection of small and medium-sized graphs, [25,26] highlight subgraphs in a result data graph with different color to identify the component of the data graph that matches a visual query. These results can be sorted by various measures such as subgraph distance (for approximate match). On the other hand, when queries are posed on a large network, in the literature there are three types of visualization strategies, namely *summarization-based*, *region-based*, and *feature-based*.

SLQ [43,45] employs a summarization-based technique to understand the results of a graph query. Specifically, given a query and a set of matches, it describes all the matches with a set of summary graphs by preserving the connectivity of the keywords associated with the graph query. This enables a user to get a big picture of all the result matches. In [19], results of a visual subgraph search are viewed in "supergraphlet-at-a-time" mode where one (super)graphlet containing result matches is displayed on the result screen one at a time. Intuitively, a supergraphlet represents a small region of a large network containing at least one match to the query. A user can explore this region and its neighborhood by double-clicking on a vertex. The work in [33] employs a *feature-based* visualization of query results in the context of weighted RDF graphs. Specifically, it provides a mechanism to explore large sets of weighted subgraphs relative to features relevant to the user.

2.5 Future Directions

While good progress has already been made, research on bridging graph querying and HCI opens up many opportunities for continued research. The final part of the tutorial presents open problems in this area. Some of these topics were introduced by recent vision papers [6,7]. Our grand vision is a pervasive desire to continue stimulating shift in our traditional thinking by bringing together HCI and graph data management to work together.

Visual querying on massive graphs. All research related to data-driven visual query interface construction, guidance for visual query formulation, visual action-aware query processing, and exploration and visualization of query results have focused either on a large set of small or medium-sized data graphs or on networks with millions of nodes. A natural extension to this paradigm is to support similar problems on massive graphs (comprising hundreds to billions of nodes), which may demand a distributed framework and novel algorithms built on top of it.

Efficient processing of complex graph queries. Current research demonstrates the viability of blending visual formulation and processing of subgraph containment and subgraph similarity search queries. It is an open problem to enhance the expressiveness of such visual querying framework to handle more complex subgraph queries such as homomorphism-based subgraph queries [14, 29], subgraph simulation [13].

Aesthetics-aware* GUI *design. An issue that is paramount to an end user but widely ignored by the data management community is the aesthetics of the layout of the GUI. Many HCI studies have asserted a strong link between visual complexity and aesthetics of web pages [41] and have attempted to measure their aesthetics automatically by analyzing HTML sources and screenshots of web pages [40]. In particular, GUI screenshot-based measure is considered superior to other methods as it better represents what a user sees [40]. The work in [34–36] proposed an array of *aesthetics metrics* to quantify visual complexity such as visual clutter, color variability, contour congestion, and layout quality.

Existing approaches to make a GUI stand out is to work out all of the aforementioned details of visual design manually. That is, the layout of a visual query interface is not automatically generated by considering various GUI *aesthetics metrics. Hence, how can we extend data-driven* GUI *construction techniques to be visual aesthetics-aware?* Note that the data-driven visual layout design problem can be reformulated as an optimization problem where the goal is to find an "optimal" layout that minimizes query formulation task complexity and visual complexity (measured using aesthetics metrics) of the interface.

Multi-faceted exploration and visualization of query results. As remarked earlier, techniques that enable rich, interactive exploration and visualization of graph query results are still in its infancy. *How can we easily explore and visualize results of a variety of subgraph queries to gain better understanding of it?* This is especially a challenging problem when the underlying graph is massive as the entire graph looks like a giant hairball and the subgraphs that are returned as results to a query are lost in the visual maze. Furthermore, it is interesting to explore additional insights that we may attach to the matched results that may enable end users for further exploration.

3. HISTORY OF THE TUTORIAL

To the best of our knowledge, this tutorial has not been presented in any major database or HCI conference. In particular, the tutorial on data exploration in [20] does not focus on graph data or bridging HCI with visual graph querying. The tutorials in [10,15,30,31,46] focus on traditional graph querying and analytics. Our tutorial is orthogonal to these efforts as we focus on visual graph querying paradigm.

4. BIOGRAPHIES

Sourav S. Bhowmick is an Associate Professor in the School of Computer Science and Engineering (SCSE), Nanyang Technological University. He leads the data management research group (DANTe) in SCSE. His research has appeared in top-tier venues in data management and analytics such as SIGMOD, VLDB, ICDE, VLDB Journal, TKDE, WWW, and KDD. Sourav has been keynote and tutorial speaker for several international conferences. He has received Best Paper Awards at ACM CIKM 2004 and ACM BCB 2011 for papers related to evolution mining and biological network summarization, respectively. His work on influence maximization was nominated for the best paper award in ACM SIGMOD 2015. Sourav has served as a PC member of premium data management and data mining conferences (*e.g.,* VLDB, ICDE, KDD, ICDM) and reviewer for various premium journals (*e.g.,* TKDE, VLDB Journal).

Byron Choi is an Associate Professor at the Department of Computer Science, Hong Kong Baptist University (HKBU). He obtained his Ph.D in Computer and Information Science from the University of Pennsylvania in 2006. His research interests include graph-structured databases, database usability, and database security. Byron's publications have appeared in premium venues such as TKDE, VLDBJ, SIGMOD, VLDB, and ICDE. He has served as a program committee member or reviewer of premium conferences and journals including PVLDB, VLDBJ, ICDE, TKDE and TOIS. He has served as the director of a Croucher Foundation Advanced Study Institute (ASI), titled "Frontiers in Big Data Graph Research" in 2015. He was a recipient of the HKBU President's Award for Outstanding Young Researcher in 2016.

Chengkai Li is an Associate Professor in the Department of Computer Science and Engineering at the University of Texas at Arlington. He received his Ph.D. degree in Computer Science from the University of Illinois at Urbana-Champaign in 2007. Chengkai's research interests are in database, data mining, Web data management, and natural language processing. He is conducting research on computational journalism, crowdsourcing and human computation, data exploration by ranking (top-k), skyline and preference queries, database testing, entity query, and usability challenges in querying graph data. Chengkai's papers have appeared in prestigious database, data mining and Web conferences (*e.g.,* SIGMOD, VLDB, CIDR, KDD, WWW, WSDM) and journals (*e.g.,* TODS, TKDD, TKDE). He has served as General Co-Chair and Program Co-Chair of IEEE IPCCC, and he has also served on the organizing committee of SIGMOD. He served on the program committees of premier conferences (*e.g.,* SIGMOD, VLDB, KDD, WWW, IJCAI). He has also been a reviewer for prestigious journals (*e.g.,* TODS, TOIS, TKDE, VLDB Journal). Chengkai is a recipient of the 2011 and 2012 HP Labs Innovation Research Award.

5. ACKNOWLEDGMENTS

Sourav S Bhowmick was supported by the Singapore-MOE AcRF Tier-2 Grant MOE2015-T2-1-040. Byron Choi was partially supported by HKRGC GRF 12232716 and 12201315. Chengkai Li was partially supported by NSF grant 1408928 and NSF-China grant 61370019. Any opinions, findings, and conclusions in this publication are those of the authors and do not necessarily reflect the views of the funding agencies.

6. REFERENCES

[1] S. Abiteboul, et al. The Lowell Database Research Self-Assessment. *In Communication of the ACM*, 2005.

[2] P. D. Adamczyk, B. P. Bailey. If Not Now, When? The Effects of Interruptions at Different Moments Within Task Execution. In *CHI*, 2004.

[3] B. P. Bailey, J. A. Konstan. On the Need for Attention Aware Systems: Measuring Effects of Interruption on Task Performance, Error Rate, and Affective State. *In Journal of Computers in Human Behavior*, 22(4), 2006.

[4] S.S. Bhowmick, H.-E. Chua, B. Thian, B. Choi. VISUAL: An HCI-inspired Simulator of Blending Visual Subgraph Query Construction and Processing. *In ICDE*, 2015.

[5] S.S. Bhowmick, C. E. Dyreson, B. Choi, M.-H Ang. Interruption-Sensitive Empty Result Feedback: Rethinking the Visual Query Feedback Paradigm for Semistructured Data. *In CIKM*, 2015.

[6] S. S. Bhowmick. DB $\bowtie$ HCI: Towards Bridging the Chasm Between Graph Data Management and HCI. *In DEXA*, 2014.

[7] S. S. Bhowmick, B. Choi, C. E. Dyreson. Data-driven Visual Graph Query Interface Construction and Maintenance: Challenges and Opportunities. *PVLDB* 9(12), 2016.

[8] H. Blau, N. Immerman, D. Jensen. A Visual Language for Querying and Updating Graphs. *Tech. Report 2002-037*, Univ. of Mass., Amherst, 2002.

[9] D.H. Chau, C. Faloutsos, H. Tong, J. I. Hong, B. Gallagher, T. Eliassi-Rad. GRAPHITE: A Visual Query System for Large Graphs. *In ICDM Workshop*, 2008.

[10] P. Cudré-Mauroux, S. Elnikety. Graph Data Management Systems for New Application Domains. *In VLDB*, 2011.

[11] A. De Angeli, A. Sutcliffe, J. Hartmann. Interaction, Usability and Aesthetics: What Influences Users' Preferences? *In Proc. of Conference on Designing Interactive Systems*, 2006.

[12] D. Erdös, Z. Fekete, A. Lukács. Visualized subgraph search. *IEEE VAST*, 2009.

[13] W. Fan, J. Li, S. Ma, N. Tang, Y. Wu, Y. Wu. Graph Pattern Matching: From Intractable to Polynomial Time. *VLDB*, 3(1-2):264–275, 2010.

[14] W. Fan, J. Li, S. Ma, H. Wang, Y. Wu. Graph Homomorphism Revisited for Graph Matching. In *PVLDB*, 2010

[15] C. Faloutsos, U. Kang. Mining Billion-Scale Graphs: Patterns and Algorithms. *In SIGMOD*, 2012.

[16] F. Haag, S. Lohmann, S. Bold, T. Ertl. Visual SPARQL Querying based on Extended Filter/flow Graphs. *In AVI*, 2014.

[17] W.-S Han, J. Lee, M.-D. Pham, J. X. Yu. iGraph: A Framework for Comparisons of Disk Based Graph Indexing Techniques. *In VLDB*, 2010.

[18] H. He, A. K. Singh. Graphs-at-a-time: Query Language and Access Methods for Graph Databases. *In SIGMOD*, 2008.

[19] H. H. Hung, S. S. Bhowmick, B. Q. Truong, B. Choi, S. Zhou. QUBLE: Towards Blending Interactive Visual Subgraph Search Queries on Large Networks. *VLDB J.* 23(3), 2014.

[20] S. Idreos, O. Papaemmanouil, S. Chaudhuri. Overview of Data Exploration Techniques. *In SIGMOD*, 2015.

[21] S. T. Iqbal, B. P. Bailey. Effects of Intelligent Notification Management on Users and Their Tasks. *In CHI*, 2008.

[22] S. T. Iqbal, B. P. Bailey. Investigating the Effectiveness of

Mental Workload as a Predictor of Opportune Moments for Interruption. *In CHI*, 2005.

[23] N. Jayaram, S. Goyal, C. Li. VIIQ: Auto-Suggestion Enabled Visual Interface for Interactive Graph Query Formulation. *In PVLDB*, 8(12), 2015.

[24] N. Jayaram, R. Bhoopalam, C. Li, V. Athitsos. Orion: Enabling Suggestions in a Visual Query Builder for Ultra-Heterogeneous Graphs. *In CoRR*, abs/1605.06856, http://arxiv.org/abs/1605.06856, 2016.

[25] C. Jin, S. S. Bhowmick, X. Xiao, J. Cheng, B. Choi. GBLENDER: Towards Blending Visual Query Formulation and Query Processing in Graph Databases. *In SIGMOD*, 2010.

[26] C. Jin, S. S. Bhowmick, B. Choi, S. Zhou. PRAGUE: A Practical Framework for Blending Visual Subgraph Query Formulation and Query Processing. *In ICDE*, 2012.

[27] U. Kang, C. Tsourakakis, C. Faloutsos. Pegasus: A Peta-scale Graph Mining System - Implementation and Observations. *In ICDM*, 2009.

[28] F. Katsarou, N. Ntarmos, P. Triantafillou. Performance and Scalability of Indexed Subgraph Query Processing Methods. *In VLDB*, 2015.

[29] A. Khan, Y. Wu, C. C. Aggarwal, X. Yan. NeMa: Fast Graph Search with Label Similarity. *VLDB*, 6(3):181–192, 2013.

[30] A. Khan, L. Chen. On Uncertain Graphs Modeling and Queries. *In VLDB*, 2013.

[31] A. Khan, S. Elnikety. Systems for Big Graphs. *In VLDB*, 2014.

[32] D. Koutra, D. Jin, Y. Ning, C. Faloutsos. PERSEUS: An Interactive Large-Scale Graph Mining and Visualization Tool. *In VLDB*, 2015.

[33] S. Liu, J. P. Cedeno, K. S. Candan, M. L. Sapino, S. Huang, X. Li. R2DB: A System for Querying and Visualizing Weighted RDF Graphs. *In ICDE*, 2012.

[34] E. Michailidou, S. Harper, S. Bechhofer. Visual Complexity and Aesthetic Perception of Web Pages. *In Proc. of ACM International Conference on Design of Communication*, 2008.

[35] A. Miniukovich, A. De Angeli. Quantification of Interface Visual Complexity. *In Working Conference on Advanced Visual Interfaces*, 2014.

[36] A. Miniukovich, A. De Angeli. Computation of Interface Aesthetics. *In SIGCHI*, 2015.

[37] C. A. Monk, J. G. Trafton, D. A. Boehm-Davis. The Effect of Interruption Duration and Demand on Resuming Suspended Goals. *J. of Experimental Psychology: Applied*, 14, 2008.

[38] P. Palanque, M. Winckler, J.-F Ladry, M. H. ter Beek, G. Faconti, M. Massink. A Formal Approach Supporting the Comparative Predictive Assessment of the Interruption-Tolerance of Interactive Systems. *In ACM EICS*, 2009.

[39] R. Pienta, A. Tamersoy, A. Endert, S. Navathe, H. Tong, D. H.Chau. VISAGE: Interactive Visual Graph Querying. *In AVI*, 2016.

[40] K. Reinecke, T. Yeh, L. Miratrix, R. Mardiko, Y. Zhao, J. Liu, K. Z. Gajos. Predicting Users' First Impressions of Website Aesthetics with a Quantification of Perceived Visual Complexity and Colorfulness. *In SIGCHI*, 2013.

[41] A. N. Tuch, E. E. Presslaber, M. Stöcklina, K. Opwis, J. A. Bargas-Avila. The Role of Visual Complexity and Prototypicality Regarding First Impression of Websites: Working Towards Understanding Aesthetic Judgements. *International J. of Human-Computer Studies*, 70, 2012.

[42] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J.J. van Wijk, J.-D. Fekete, D.W. Fellner. Visual Analysis of Large Graphs: State-of-the-art and Future Research Challenges. *Computer Graphics Forum*, 2011.

[43] Y. Wu, S. Yang, M. Srivatsa, A. Iyengar, X. Yan. Summarizing Answer Graphs Induced by Keyword Queries. *VLDB*, 6(13), 2013.

[44] X. Xie, Z. Fan, B. Choi, P. Yi, S. S. Bhowmick, S. Zhou. PIGEON: Progress Indicator for Subgraph Queries. *ICDE*, 2015.

[45] S. Yang, Y. Xie, Y. Wu, T. Wu, H. Sun, J. Wu, X. Yan. SLQ: A User-friendly Graph Querying System. *In SIGMOD*, 2014.

[46] D. Yan, Y. Bu, Y. Tian, A. Deshpande, J. Cheng. Big Graph Analytics Systems. *In SIGMOD*, 2016.

[47] P. Yi, B. Choi, S. S. Bhowmick, J. Xu. AutoG: A Visual Query Autocompletion Framework for Graph Databases. *In The VLDB Journal*, 2017.

[48] J. Zhang, S. S. Bhowmick, H. H. Nguyen, B. Choi, F. Zhu. DAVINCI: Data-driven Visual Interface Construction for Subgraph Search in Graph Databases. *In IEEE ICDE*, 2015.