V

Visual Graph Querying

Sourav S. Bhowmick[1] and Byron Choi[2]
[1]Nanyang Technological University, Singapore, Singapore
[2]Hong Kong Baptist University, Hong Kong, China

Definitions

Visual querying of graphs: Formulation of a graph query by drawing it using a visual query interface.

Overview

Querying graphs has emerged as an important research problem due to the prevalence of graph-structured data in many real-world applications (e.g., social networks, road networks, collaboration networks, cheminformatics, bioinformatics, computer vision). At the core of many of these applications lies a common and important query primitive called *subgraph search*, which retrieves one or more matching subgraphs or data graphs containing *exact* (Han et al. 2013; Yan et al. 2004) or *approximate* (Yan et al. 2005; Tian and Patel 2008) match of a user-specified query graph (aka subgraph query). Since the last decade, a number of graph query languages (e.g., SPARQL, *Cypher*) have been proposed to facilitate textual formulation of subgraph queries. All these languages assume that a user has programming and debugging expertise to formulate queries correctly in these languages. Unfortunately, this assumption makes it harder for nonprogrammers to take advantage of a subgraph search framework as it requires significant time and effort to learn these languages. For example, domain experts such as chemists cannot be expected to learn the complex syntax of a graph query language in order to formulate meaningful queries over a chemical compound database such as *PubChem* (http://pubchem.ncbi.nlm.nih.gov/) or *eMolecule* (https://www.emolecules.com/). Hence, it is important 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 (aka GUI) that can enable an end user to draw a subgraph query interactively have gained increasing attention in recent times from both academia and industry (e.g., *PubChem*, *eMolecule*, *DrugBank*). 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 and interactive guidance to users by providing relevant and timely feedback and suggestions during query formulation, exploration, and visualization of graph query results, among others (Bhowmick et al. 2017a).

S. Sakr, A. Zomaya (eds.), *Encyclopedia of Big Data Technologies*,
https://doi.org/10.1007/978-3-319-63962-8_78-1

This chapter gives an overview to the topic of visual graph querying, discussing the state of the art in the industry and in the academic world. In particular, we emphasize research efforts that aim to bridge two traditionally disparate topics in computer science, namely, graph querying and human-computer interaction (HCI). In the next section, we present an overview of the key problems and solutions in the arena of visual graph querying. Next, in section "An Example" we present an example to illustrate them. The last section concludes this chapter by highlighting future research directions.

Key Research Findings

Our discussion on research findings follows a top-down approach, starting from visual query formulation, proceeding to visual query processing, and concluding with exploration of query results.

Visual Query Formulation

There are mainly three popular approaches for formulating visual subgraph queries. In the *edge-at-a-time* approach (e.g., Jin et al. 2012), a query is incrementally constructed by adding one edge at a time. Note that it may be time-consuming to formulate a query with a large number of edges using this approach. Hence, in the *pattern-at-a-time* approach, one may compose a visual query by dragging and dropping canned patterns or subgraphs (e.g., benzene, chlorobenzene patterns) that are available on the GUI in addition to single edge construction. Figure 1 depicts an example of such an interface. Observe that this approach is more efficient than the former as it typically takes lesser time to construct a query. For instance, instead of drawing six edges incrementally to construct a benzene ring in a query, we can construct it with a single click and drag if it is available as a canned pattern.

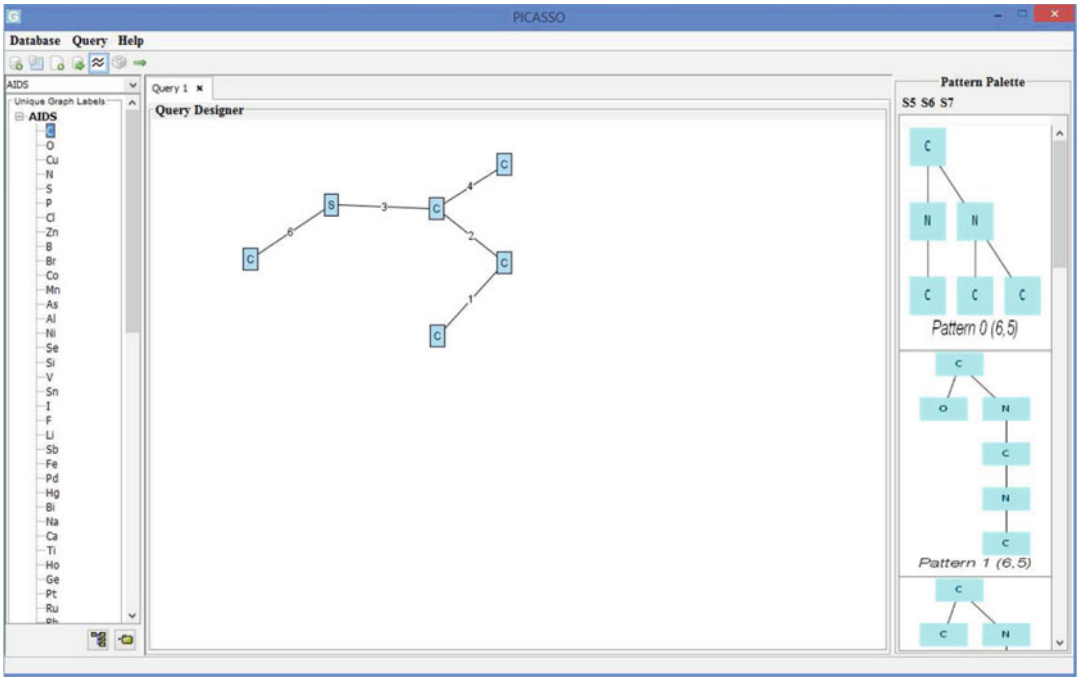
Furthermore, the pattern-at-a-time approach can be either *static* or *dynamic* in nature. In the former case, the set of patterns is fixed and displayed in the GUI (e.g., Fig. 1). In the latter case, the patterns are dynamically suggested during visual query formulation by utilizing the knowledge of partially formulated query fragments. The AUTOG (Yi et al. 2017) framework adopts this dynamic strategy. Specifically, it automatically generates a small list of subgraph as suggestions by considering potential query result size as well as structural diversity. Figure 2 shows an example of subgraph suggestions during query formulation in AUTOG. Note that it is extremely difficult to speculate a priori the subgraph structure that a user wishes to construct during query formulation.

Lastly, recent frameworks such as VISAGE (Pienta et al. 2016) support *query by example* (QBE) for formulating graph queries using examples.

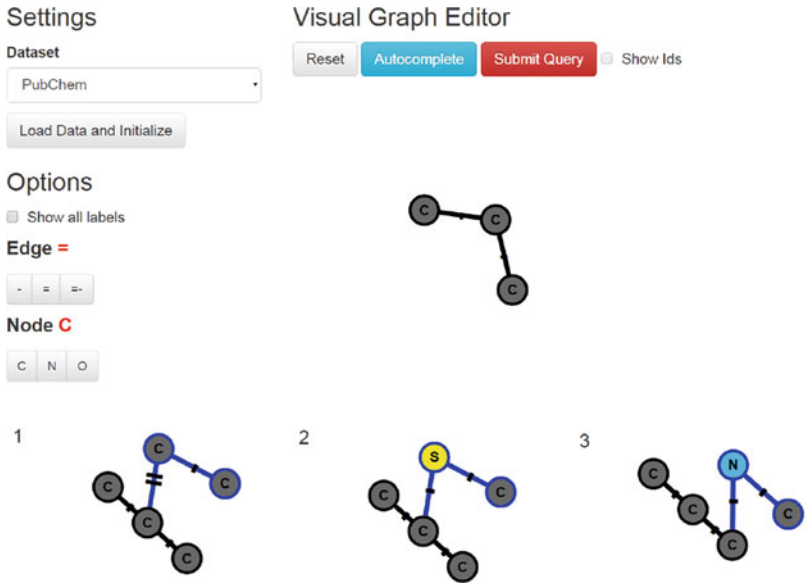
Recently, there has been another line of work that provides opportune *empty result feedback* during visual query formulation. Specifically, the goal is to detect during query construction whether a partially constructed query returns empty results or alerts users in a timely fashion so that one can undertake appropriate remedial action(s). Several recent studies (Bhowmick et al. 2015; Pienta et al. 2016) address this problem by notifying a user opportunely when a partially constructed visual query yields an empty result.

Blending Visual Query Formulation and Processing

In traditional visual query processing paradigm, query evaluation can be performed in two key steps. First, the visual query is transformed into its textual or algebraic form. Second, the transformed query is evaluated using an existing state-of-the-art graph query processing method (Han et al. 2013). Observe that although the final query that a user intends to pose is revealed gradually in a step-by-step manner during query construction, it is not exploited by the query processor prior to clicking of the `Run` icon to execute the query. This often results in slower *system response time* (SRT), which is the duration between the time a user presses the `Run` icon and the time when the user gets the query results (Jin et al. 2012). This traditional view of visual graph query processing is primarily due to the fact that until recently the data management community has traditionally



Visual Graph Querying, Fig. 1 Pattern-at-a-time visual query formulation



Visual Graph Querying, Fig. 2 Subgraph suggestions during query formulation

considered visual interface-related issues more relevant to the HCI community and orthogonal to data processing.

The techniques in Jin et al. (2010, 2012) and Hung et al. (2014) take a nontraditional step toward exploring a graph query processing paradigm by blending these two orthogonal areas. Specifically, it *interleaves* (i.e., *blend*) query construction and query processing to prune false results and prefetch partial query results in a single-user environment by exploiting the availability of GUI *latency* (i.e., the time taken

to construct an edge of a query graph visually) during visual query formulation. The key benefits of this paradigm are at least two-fold. First, it significantly improves the SRT (From an end user's perspective, the SRT is crucial as it is the time a user has to wait before she can view the results.) as the query processor does not remain idle during visual query formulation by processing the potential subgraph query early based on "hints" received from the user. Second, as a visual query is iteratively processed during query formulation, it paves way for realizing efficient techniques that can enhance usability of graph databases such as query suggestion and exploratory search (Huang et al. 2017).

This framework constructs offline and online indexes based on underlying features of data graphs. When a user adds a new edge e to a visual subgraph query q, it constructs and maintains an adaptive online index for the edge to facilitate blending of visual query formulation and processing. If the user is interested in exact subgraph matches, then it retrieves the candidates of q (stored in R_q) by leveraging the offline and online indexes. If R_q is empty, then it means that there is no exact match for q after the addition of e. Consequently, the user can either modify q or retrieve similar matches to q. If the user chooses the latter, then q is regarded as a subgraph similarity query, and corresponding candidate matches are retrieved by leveraging the indexes. On the other hand, if the user chooses the former, then a user-selected edge is removed and the online index is updated. If the user clicks the `Run` button, then the constructed query q is processed to retrieve result matches . If q is an exact subgraph query, the exact results will be returned after conducting candidates verification (i.e., subgraph isomorphism test), if necessary, on R_q. Otherwise, if it is already a subgraph similarity query, then results that match q approximately are returned to the user.

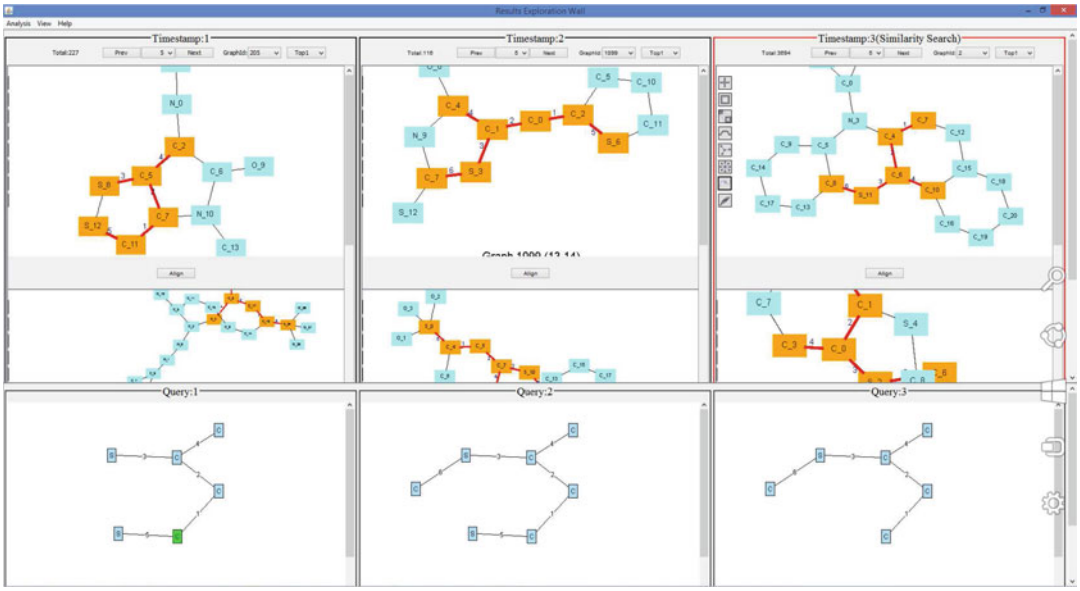
Note that in the above framework, the expensive candidate verification step is performed only after the `Run` button is clicked and not during visual construction of the query fragments. Second, it allows a user to execute a query fragment anytime during query formulation and not wait until the entire query is visually formulated. This facilitates exploratory search as a user may formulate a partial query, execute it, browse the results, and then continue expanding it (Huang et al. 2017).

Interactive Exploration of Query Results

The preceding subsection highlights frameworks that exploit human interactions with a visual graph query interface during query formulation to process subgraph queries iteratively. Naturally, it is imperative for such visual subgraph querying frameworks to enable efficient exploration and visualization of result matches of such subgraph queries. This is a challenging problem as it requires effective summarization and visualization of the content and structure of the matching subgraphs involved in a potentially large collection of query results. In this subsection, we briefly describe efforts reported in recent literature to this end.

Despite the fact that exploration of query results is the first step toward sensemaking for a user, scant attention has been paid on this problem by both academic and commercial graph data management communities. Specifically, we can categorize visual exploration of graph query results into two types: (a) visual exploration of query results in a large collection of small- and medium-sized graphs (Jin et al. 2010, 2012; Huang et al. 2017) and (b) visual exploration of result matches in a large network (Hung et al. 2014; Pienta et al. 2016, 2018). We elaborate on them in turn.

Exploration in a large collection of small- and medium-sized graphs. Early efforts for exploration of query results (Jin et al. 2010, 2012) simply displayed result matches in a list, without revealing connections among results. A matching subgraph in each data graph is then highlighted with different colors to identify the component of the data graph that matches a visual query. These results may be further sorted by various measures such as subgraph distance (for approximate match). For instance, in Jin et al. (2010, 2012), a user can only iteratively scroll through each result data graph to view query results. Clearly, this is



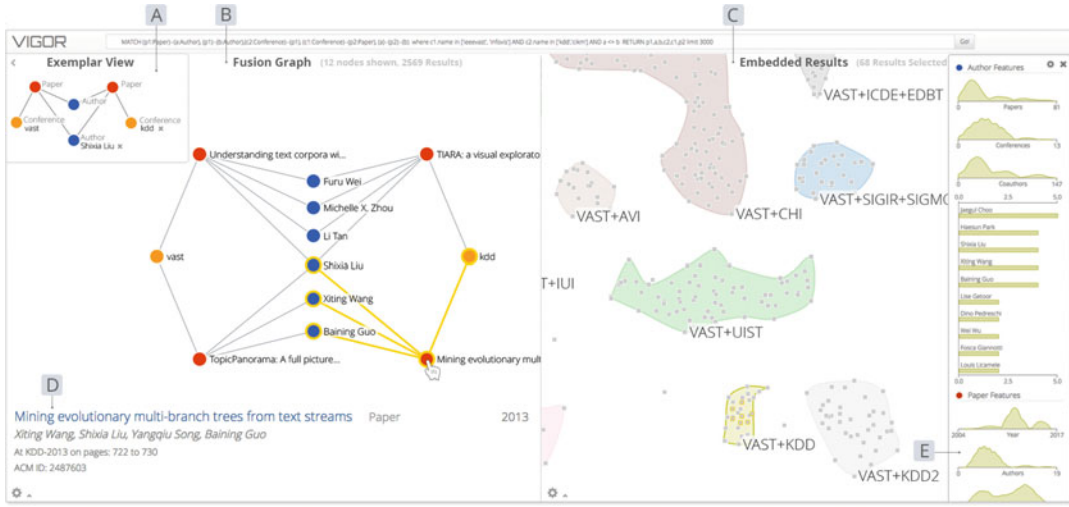
Visual Graph Querying, Fig. 3 Parallel search streams in PICASSO

tedious even for modest-sized query results and cannot easily reveal patterns and relationships between matching subgraphs.

PICASSO (Huang et al. 2017) is a system that utilizes the query processing engine described in the preceding subsection to support visual exploratory search. It accommodates the results of the initial and reformulated query graphs to be juxtaposed in the form of *parallel search streams* (i.e., parallel query-results pairs) that facilitate exploration of the underlying data and possible identification of new search directions. Figure 3 depicts an example of three parallel search streams during exploratory search in PICASSO. Furthermore, it provides a framework to further search and analyze various features of the search results during the exploration process to facilitate understanding of the data.

Exploration of result matches in a large network. It is well known that visualizing large graphs containing thousands of vertices and edges is cognitively challenging. Consequently, the aforementioned approach of highlighting the result matches of a visual subgraph query by color-coding them in the original data graph is ineffective in the context of large networks as it is not only challenging to locate query results in a giant "hair ball" but also it is extremely difficult to comprehend the structural relationships among the vertices in a matching subgraph overlaid on a large network. Despite these challenges, there are very few works on query results exploration on large networks. Similar to Jin et al. (2010) and Jin et al. (2012), early efforts such as VISAGE (Pienta et al. 2016) simply display the results in form of a list.

Recently, there has been an increasing attention to develop advanced techniques for query results exploration for large networks. This work can be broadly classified into three types, namely, *region-based*, *exemplar-based*, and *feature-based exploration*. Intuitively, a *region-based exploration* scheme iteratively displays a small region of the underlying network containing a result match of a subgraph query. By showing only a fragment of the original network one at a time, it alleviates the cognitive overhead associated with the visualization of all query results on the original network. The approach in Hung et al. (2014) adopts this strategy. In an *exemplar-based exploration* scheme, a user can select a specific query result (i.e., an exemplar) and relax its constraints to retrieve other similar



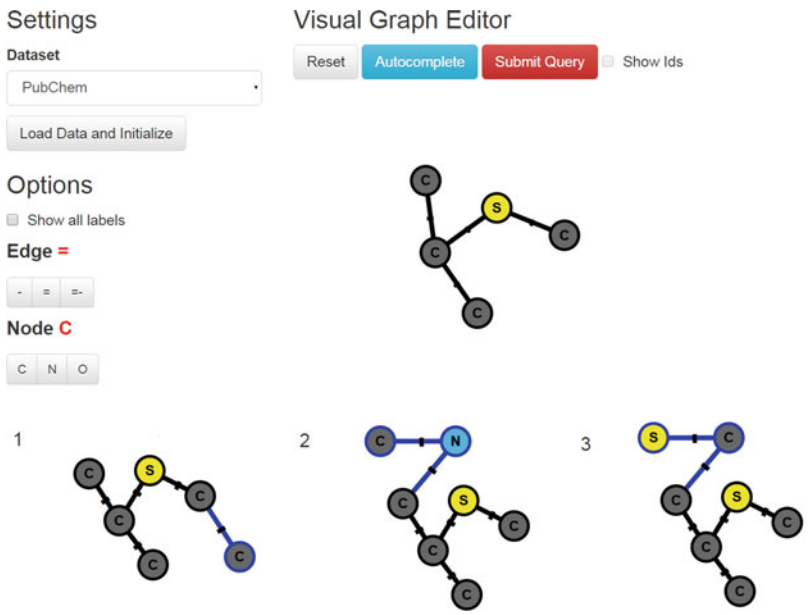
Visual Graph Querying, Fig. 4 Visual interface of VIGOR (Pienta et al. 2018)

results. A user can also start the exploration by specifying only the topology (without constraints on node values) and iteratively add constraints to narrow in on specific results. *Feature-based exploration* schemes, on the other hand, take a top-down approach by generating a high-level overview of all the query results. Specifically, it groups the query results based on the structural features and embeds them in a low-dimensional representation. VIGOR (Pienta et al. 2018) supports both exemplar-based and feature-based exploration schemes (Fig. 4).

An Example

Consider a chemical compound repository containing a large collection of small- or medium-sized graphs such as *PubChem*. Suppose a user wishes to formulate the subgraph query shown in Fig. 1. She may visually formulate it either using an interface that supports edge-at-a-time query formulation by adding the five edges iteratively or using a static pattern-at-a-time interface (e.g., Fig. 1) to quicken the formulation by dragging and dropping canned patterns. Alternatively, a dynamic pattern-at-a-time interface such as Fig. 2 can be leveraged where suggestions are generated iteratively as the user formulates her query. Specifically, Fig. 2 depicts the three suggestions generated by AUTOG (Yi et al. 2017) after the addition of two C-C edges. The query is shown in the middle of the GUI, whereas relevant suggestions are presented in the bottom. She may adopt the second suggestion and continue with the query formulation task. Next, AUTOG presents the user's final query as Suggestion 3 as shown in Fig. 5. Consequently, the selection of this suggestion completes the query formulation task.

The techniques proposed in Jin et al. (2010, 2012) can be utilized to interleave aforementioned query formulation with query processing. Specifically, the query can be iteratively executed during formulation, and corresponding results can be explored to gain insights. Figure 3 depicts such exploration. The original query is shown in the bottom left panel. The modifications to this query with addition and deletion of edges are shown in the bottom middle and right panels, respectively. The corresponding search results for each of these subgraph queries are shown in the top. A user can compare the search results and explore the differences between the result sets and features of the nodes in them using PICASSO (Huang et al. 2017).



Visual Graph Querying, Fig. 5 Query suggestions of Suggestion 2 of Fig. 2, where Suggestion 3 is the completed query

Future Directions

While good progress has already been made, research on visual graph querying opens up many opportunities for continued research as highlighted below. Some of these topics were introduced by recent vision papers (Bhowmick 2014; Bhowmick et al. 2016).

Data-driven construction of visual query interfaces. Most of the real-world GUIs for visual graph querying 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 to aid efficient query formulation, are static in nature when the underlying graph data repository evolves, and have limited portability (Bhowmick et al. 2016). To alleviate these limitations, a recent work introduced the paradigm of *data-driven* GUI construction and maintenance (Zhang et al. 2015), where the goal is to 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. However, these are very early efforts, and there are many opportunities to explore this novel paradigm of GUI construction further.

Visual querying on massive graphs. All researches related to 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 (Fan et al. 2010).

Multifaceted 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 hair ball 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.

Automated performance study. User studies are *sine qua non* for evaluating the performance and effectiveness of the aforementioned visual action-aware query processing techniques. 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 (Katsarou et al. 2015), 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, it is important to 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*. A recent effort (Bhowmick et al. 2017b) addresses this by building an HCI-based framework for simulating subgraph query construction on a database containing a large number of small- or medium-sized graphs. It can be used to automate exhaustive evaluation of performances of various visual action-aware query processing techniques. However, this area is still in its infancy with only one notable work.

Cross-References

- Graph Exploration and Search
- Graph Pattern Matching
- Graph Query Languages

References

Bhowmick SS (2014) DB ⋈ HCI: towards bridging the chasm between graph data management and HCI. In: Proceedings of the 25th international conference on database and expert systems applications (DEXA), Part I, Munich, 1–4 Sept 2014, pp 1–11

Bhowmick SS, Dyreson CE, Choi B, Ang M (2015) Interruption-sensitive empty result feedback: rethinking the visual query feedback paradigm for semistructured data. In: Proceedings of the 24th ACM international conference on information and knowledge management (CIKM), Melbourne, 19–23 Oct 2015, pp 723–732

Bhowmick SS, Choi B, Dyreson CE (2016) Data-driven visual graph query interface construction and maintenance: challenges and opportunities. PVLDB 9(12):984–992

Bhowmick SS, Choi B, Li C (2017a) Graph querying meets HCI: state of the art and future directions. In: Proceedings of the 2017 ACM international conference on management of data, SIGMOD conference 2017, Chicago, 14–19 May 2017, pp 1731–1736

Bhowmick SS, Chua H, Choi B, Dyreson CE (2017b) VISUAL: simulation of visual subgraph query formulation to enable automated performance benchmarking. IEEE Trans Knowl Data Eng 29(8):1765–1778

Fan W, Li J, Ma S, Wang H, Wu Y (2010) Graph homomorphism revisited for graph matching. PVLDB 3(1):1161–1172

Han W, Lee J, Lee J (2013) $\text{Turbo}_{\text{iso}}$: towards ultrafast and robust subgraph isomorphism search in large graph databases. In: Proceedings of the ACM SIGMOD international conference on management of data (SIGMOD), New York, 22–27 June 2013, pp 337–348

Huang K, Bhowmick SS, Zhou S, Choi B (2017) PICASSO: exploratory search of connected subgraph substructures in graph databases. PVLDB 10(12):1861–1864

Hung HH, Bhowmick SS, Truong BQ, Choi B, Zhou S (2014) QUBLE: towards blending interactive visual subgraph search queries on large networks. VLDB J 23(3):401–426

Jin C, Bhowmick SS, Xiao X, Cheng J, Choi B (2010) GBLENDER: towards blending visual query formulation and query processing in graph databases. In: Pro-

ceedings of the ACM SIGMOD international conference on management of data, (SIGMOD), Indianapolis, 6–10 June 2010, pp 111–122

Jin C, Bhowmick SS, Choi B, Zhou S (2012) PRAGUE: towards blending practical visual subgraph query formulation and query processing. In: IEEE 28th international conference on data engineering (ICDE), Washington, DC, 1–5 Apr 2012, pp 222–233

Katsarou F, Ntarmos N, Triantafillou P (2015) Performance and scalability of indexed subgraph query processing methods. PVLDB 8(12):1566–1577

Pienta R, Tamersoy A, Endert A, Navathe SB, Tong H, Chau DH (2016) VISAGE: interactive visual graph querying. In: Proceedings of the international working conference on advanced visual interfaces (AVI), Bari, 7–10 June 2016, pp 272–279

Pienta R, Hohman F, Endert A, Tamersoy A, Roundy K, Gates C, Navathe SB, Chau DH (2018) VIGOR: interactive visual exploration of graph query results. IEEE Trans Vis Comput Graph 24:215–225

Tian Y, Patel JM (2008) TALE: a tool for approximate large graph matching. In: Proceedings of the 24th international conference on data engineering (ICDE), Cancún, 7–12 Apr 2008, pp 963–972

Yan X, Yu PS, Han J (2004) Graph indexing: a frequent structure-based approach. In: Proceedings of the ACM SIGMOD international conference on management of data, Paris, 13–18 June 2004, pp 335–346

Yan X, Yu PS, Han J (2005) Substructure similarity search in graph databases. In: Proceedings of the ACM SIGMOD international conference on management of data, Baltimore, 14–16 June 2005, pp 766–777

Yi P, Choi B, Bhowmick SS, Xu J (2017) Autog: a visual query autocompletion framework for graph databases. VLDB J 26(3):347–372

Zhang J, Bhowmick SS, Nguyen HH, Choi B, Zhu F (2015) Davinci: data-driven visual interface construction for subgraph search in graph databases. In: 31st IEEE international conference on data engineering (ICDE), Seoul, 13–17 Apr 2015, pp 1500–1503