Data-driven Visual Query Interfaces for Graphs: Past, Present, and (Near) Future

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

Byron Choi
Hong Kong Baptist University
Hongkong SAR, China
bchoi@comp.hkbu.edu.hk

ABSTRACT

Visual graph query interfaces (VQI) widen the reach of graph querying frameworks across a variety of end users by enabling non-programmers to use them. Several industrial and academic frameworks for querying graphs expose such visual interfaces. In this tutorial, we survey recent developments in the emerging area of *data-driven* visual query interface that is grounded on the principles of human-computer interaction (HCI) and cognitive psychology to enhance usability of graph querying frameworks. A data-driven VQI has many benefits such as reducing the cost in constructing and maintaining an interface, superior support for query formulation, and increased portability of the interface. We discuss the notion of making VQIs data-driven and compare it with its classical manual counterpart, and review techniques for automatic construction and maintenance of these interfaces. 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 data-driven visual graph query interface management, giving researchers a snapshot of the current state of the art in this topic, and future research directions.

CCS CONCEPTS

• **Information systems** → *Query languages for non-relational engines.*

KEYWORDS

Visual query interface, graph search, data-driven, cognitive load

ACM Reference Format:
Sourav S Bhowmick and Byron Choi. 2022. Data-driven Visual Query Interfaces for Graphs: Past, Present, and (Near) Future. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3514221.3522562

1 INTRODUCTION

"If the user can't use it, it doesn't work."

Susan Dray
Dray & Associates, Inc.

Graphs are ubiquitous nowadays in many application domains (*e.g.,* biology, social sciences, chemistry, finance). The global graph database market size is expected to grow from USD 1.9 billion in 2021 to USD 5.1 billion by 2026 [5]. However, according to a recent *Markets and Markets* report [5], a potential bottleneck for this growth is that *"developers have to write their queries using Java as there is no Standard Query Language (SQL) to retrieve data from graph databases, which means employing expensive programmers or developers have to use SPARQL or one of the other query languages that have been developed to support graph databases, however, it would mean learning a new skill. This results in the lack of standardization and programming ease for graph database systems."*. This is also echoed by a recent survey [41] which revealed that graph query languages and usability are considered as some of the top challenges for graph processing. Although considerable efforts have been invested toward efficient and scalable processing of graphs [15, 43], the above issues have received relatively lesser attention from the data management community for decades.

Fortunately, graphs are more intuitive to draw than to compose them in textual format. Hence, a common starting point for addressing these challenges is the deployment of a visual query interface (VQI) that can enable an end user to draw a graph query interactively instead of formulating it textually using a graph query language. Indeed, several industrial graph querying frameworks in a variety of domains provide such interfaces [2–4, 6]. These VQIs typically utilize *direct-manipulation* interfaces that are appealing to *"novices as they can learn basic functionality quickly, are easy to remember for intermittent users by retaining operational concepts, and can be rapid for frequent users"* [42]. Interestingly, majority of these VQIs do not expose interfaces to formulate queries using a graph query language, highlighting the reluctance of many domain-specific end users to use such programming languages. Given that query formulation *precedes* query processing, VQIs naturally facilitate democratization of graph querying frameworks by empowering end users with no programming background to formulate subgraph queries and thereby exploit powerful graph query processing engines for their tasks. Indeed, a powerful query processor has no practical usage to an end user if he/she fails to formulate subgraph queries to express his/her search goals!

The construction of existing VQIs is typically performed by programmers *manually* coding various interface features based on domain knowledge that may be provided by domain experts. Such endeavour, however, is labour-intensive and may demand a comprehensive knowledge of different topological structures in the underlying graph data for superior VQI design [24, 48]. Furthermore, these VQIs lack of portability across different graph data sources and applications as several components of a VQI need to be

SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA

ACM ISBN 978-1-4503-9249-5/22/06…$15.00
https://doi.org/10.1145/3514221.3522562

Table 1: Tutorial overview.

Topic	*Approx. time (min)*	*Representative papers*	*Demo*	*Code*
Introduction	5	-	No	-
Usability of manual VQI	15	[2–4, 6, 16, 20, 21, 26, 38, 47]	Yes ([6, 26])	-
The concept of data-driven VQI	10	[7, 10]	No	-
Data-driven construction of VQIs	30	[12, 24, 45, 48, 51]	Yes ([12, 49, 51])	https://github.com/MIDAS2020/CATAPULT
Data-driven maintenance of VQIs	10	[25]	Yes ([12])	https://github.com/MIDAS2020/Midas
Future research direction	15	-	No	-

reconstructed due to their data dependency [10]. The maintenance of these components as the underlying data source evolves becomes challenging as well [25]. Given that industrial and academic entities will continue to generate disparate graph data sources for different applications, a *data-driven* approach to VQI generation and maintenance is paramount. Such a data-driven paradigm gives end users the freedom to easily and quickly construct and maintain a VQI for any data sources without resorting to coding. This has recently paved the way for research in data-driven construction and maintenance of visual graph query interfaces [10, 12, 24, 25, 45, 48, 51].

This tutorial gives a comprehensive introduction to the topic of data-driven visual graph query interface management. A hallmark of this tutorial is to emphasize research efforts that aim to bridge three traditionally orthogonal topics or fields, namely, graph querying, human computer interaction (HCI), and cognitive psychology. Specifically, a key component we cover is the review of techniques and strategies that make data-driven visual graph query interface construction and maintenance grounded on the principles of HCI and cognitive psychology to augment usability of graph querying frameworks. An overview of the scope of the tutorial is as follows.

- **Usability of manual VQI:** We begin by reviewing different features of existing manually-constructed VQIs and analyze their usability. Specifically, we emphasize them from three dimensions that influence *usability criteria*, namely *search paradigm*, *maintainability*, and *aesthetics*.
- **Data-driven VQI:** We give an overview of the notion of *data-driven* VQI and compare it with its manual counterpart based on the aforementioned features, highlighting its strengths w.r.t usability and portability. Then we articulate the challenges to realize it.
- **Data-driven construction of a VQI:** In this part, we review HCI- and cognitive psychology-aware techniques for constructing a VQI *automatically* from a given data source. Specifically, we present techniques that are designed to handle a large collection of small- or medium-sized data graphs (*e.g.,* chemical compounds) and large networks (*e.g.,* biological networks, social networks).
- **Data-driven maintenance of a VQI:** We review a technique proposed in the literature to automatically maintain the relevant components of a data-driven VQI as the underlying data graphs evolve. Specifically, we highlight efforts to make such maintenance cognitive psychology-conscious.
- **Future research directions:** Finally, we discuss open problems on the topic of data-driven VQI management. We discuss how the data-driven paradigm on graphs paves the way for similar research endeavour on other data types (*e.g.,* time series). Lastly, we highlight the usefulness of the techniques presented here in addressing problems beyond VQI construction (*e.g.,* graph summarization).

Table 1 shows the approximate duration of each of the above topics for a 90-min tutorial. A detailed treatment of these topics appear in [7].

2 TUTORIAL OUTLINE

Our presentation follows a top-down approach, starting from the usability of existing manual visual graph query interfaces (VQIs), proceeding to the introduction of the notion of data-driven VQI and its potential to enhance usability, construction and maintenance of a data-driven VQI, and concluding with future research directions in this arena. Table 1 shows the key papers we discuss. Additionally, a key feature of our tutorial is the live demonstrations of various academic and industrial VQIs related to the aforementioned topics as highlighted in the fourth column. We also make the codebase available for selected data-driven VQI frameworks.

2.1 Usability of Manual Visual Query Interfaces

Visual query interfaces for graph-structured data (VQI) have been used in academia and industry for more than a decade [2–4, 6, 16, 20, 21, 26, 27, 30, 38, 39, 47]. The key components in majority of these VQIs are the *Attribute Panel*, *Query Panel*, *Pattern Panel*, and *Results Panel*. The *Attribute* and *Pattern* panels are optional components containing attributes of nodes or edges and small connected graphs, respectively. The *Query Panel* is used by a user to draw a query and the *Results Panel* is for visualizing the query results. The details of visual design and the contents of some of these components are created manually by "hard coding" them during implementation of a VQI. We refer to these VQIs collectively as *manual* VQIs. These interfaces enable an end user to visually formulate a variety of graph queries (*e.g.,* subgraph matching, subgraph similarity, subgraph enumeration). A common thread across all these different query types is the visual construction of a graph topology.

We begin our tutorial by analyzing manual VQIs w.r.t their *usability*. Usability is a quality attribute that assesses how easy user interfaces are to use [8]. Specifically, it refers to the *"quality of the interaction in terms of parameters such as time taken to perform tasks, number of errors made, and the time taken to become a competent user"* [8]. To this end, we first compare these VQIs w.r.t *search paradigm*, *maintainability*, and *aesthetics* as they influence various *criteria* of usability. We elaborate on them in turn.

Search paradigms. Typically, an end user follows a *top-down* or *bottom-up* visual search paradigm for formulating queries. The former refers to searching graph data based on a user's intuition of

how the desired query should look like in theory (*i.e.,* translating a graph topology "in-the-head" to a visual query). One is expected to possess a precise knowledge about the attributes in the underlying graph repository as well as topologies-of-interest to formulate meaningful queries. The latter refers to search when a user does not have an upfront knowledge of the complete query structure. That is, she may have some concepts or keywords in-the-head but is unaware of how they form a connected query graph structure that may result in a meaningful query. Hence, she may get acquainted to the key substructures that exist in the dataset through *representative objects* to galvanize query formulation. Clearly, any superior VQI should facilitate both these search paradigms. In particular, support for bottom-up search is paramount as a large network looks like a "hairball" on a visual interface and hence it is cognitively challenging to browse it to figure out which topological patterns one can use to trigger a query formulation task. Even for a database of small- or medium-sized data graphs, it is tedious to manually browse thousands of data graphs to seek for topologies of interest to kick-start a search. Although all existing manual VQIs support top-down search, very few [3, 4, 6, 26] provide functionalities to support effective bottom-up search. Particularly, a *Pattern Panel* containing *patterns* is exposed to an end user as representative objects to kick-start query formulation. Intuitively, a pattern is a small connected subgraph (*e.g.,* benzene ring, triangle, rectangle) occurring in the underlying data graphs that one can utilize to visually construct a query graph in *pattern-at-a-time* mode.

Maintainability of a VQI. A majority of real-world graphs are dynamic in nature. For example, a study [52] reported that approximately 4,000 new structures were added daily to the *SCI finder* database (www.cas.org/products/scifinder). New compounds are also added to *PubChem* (pubchemdocs.ncbi.nlm.nih.gov/submissions-getting-started) and *Drugbank* (dev.drugbank.com/guides/faqs) daily. Similarly, large networks such as coauthorship networks (*e.g.,* DBLP) and social networks (*e.g.,* Twitter) evolve with time. We observe that all existing manual VQIs exhibit poor and inefficient *maintainability* with the evolution of underlying data graphs. Consequently, contents of various components of a VQI (*e.g., Pattern Panel*) may grow stale quickly over time, adversely impacting visual query formulation [25].

Aesthetics. Lastly, people prefer attractive interfaces [18]. The visual appearance of a VQI (*i.e.,* aesthetics) impacts its usability as it influences the way users interact with it (*i.e., aesthetic-usability effect* [1]. Specifically, the characteristics (*e.g.,* size, color) of various panels in a VQI influence its *complexity*. A visual pattern can be considered complex if its components are difficult to identify and separate from each other [37]. Several studies in HCI and psychology have found a strong relationship between aesthetic preferences and visual complexity [9, 40]. According to Berlyne's aesthetic theory [9], the relationship between them follows an inverted U-shaped curve where stimuli of a moderate degree of visual complexity is considered pleasant but both less and more complex stimuli are considered unpleasant. We report that several existing VQIs (*e.g.,* [6]) suffer from poor aesthetics. In particular, we observe that supporting the aforementioned search paradigms and query formulation-friendly functionalities may adversely impact aesthetics and visual complexity if VQIs are not designed carefully.

Usability criteria. The aforementioned three features impact several of the following criteria for usability [19]:

- *Learnability:* By which new users can interact effectively and achieve maximal performance;
- *Flexibility:* Multiple ways a user and the system exchange information;
- *Robustness:* The level of support provided to a user in determining successful achievement and assessment of goals;
- *Efficiency:* Once a user learns about a system, the speed with which he/she can perform tasks;
- *Memorability:* How easily a user will remember a system's functions, after not using it for a period;
- *Errors:* It is about the number of errors made by users, their severity, and whether they can recover from them easily;
- *Satisfaction:* How enjoyable and pleasant is it to work with the system?

Specifically, search paradigms influence flexibility, robustness, efficiency, errors, and satisfaction. Maintainability impacts efficiency and robustness whereas aesthetics contributes to satisfaction. For instance, exposing patterns in a *Pattern Panel* and maintaining them enable end users to formulate queries efficiently in multiple ways (pattern-at-a-time as well as edge-at-a-time modes) [24, 25, 48] and may reduce errors compared to visual formulation without the aid of such patterns. Furthermore, facilitating both top-down and bottom-up search by exploiting the *Pattern* and *Attribute Panels* potentially increase user satisfaction. In particular, HCI research shows that users may become frustrated if a large number of small atomic actions (*e.g.,* repeated edge construction) is necessary to accomplish a higher-level task (*e.g.,* subgraph query formulation) [42]. Naturally, patterns may ease such frustration.

A key point of our discussion is the emphasis on the fact that despite decades of research by the HCI community related to usability and human factors, many of the manual VQIs for graph querying are oblivious to these results.

2.2 Data-driven Visual Query Interfaces

The aforementioned manual VQIs adversely impact various usability criteria such as flexibility, robustness, efficiency, and satisfaction. Specifically, they do not provide sufficient features to aid flexible and efficient visual query formulation, and are static in nature when the underlying graph repository evolves [10]. Furthermore, the manual construction of a VQI limits its portability across different domains and sources as one has to reimplement and customize the VQI for each one of them [10]. Next, we introduce the notion of *data-driven* VQI [10, 12, 51] that is recently proposed to alleviate these limitations, highlight its features and advantages in comparison to manual VQIs, and challenges to realize it.

A data-driven VQI takes a fundamentally different approach in VQI construction. Given a graph repository D and a certain *budget*, it *automatically* populates and maintains various panels of a VQI from D consistent with the budget. Observe that in a VQI the contents of the *Query* and *Result Panels* are dependent on users and user-specified queries, respectively. On the other hand, contents of the *Attribute* and *Pattern Panels* hinge on D. Hence, data-driven VQIs automatically populate the contents of these two panels from D to aid visual query formulation. These contents influence several

usability criteria (*e.g.,* efficiency, flexibility, satisfaction). Consequently, such a data-driven paradigm brings in several benefits such as superior support for visual subgraph query construction, significant reduction in the cost of constructing and maintaining a VQI, and portability of a VQI across diverse graph data source and querying applications [24, 25, 45, 48]. We believe that as the number of graph data sources grows with time, data-driven VQIs will offer sufficient benefits to developers and end users of graph repositories by making generation of usable VQIs effortless.

2.3 Construction of Data-driven VQIs

Next, we focus on frameworks for constructing data-driven VQIs from graph repositories [24, 45, 48, 51]. Observe that attributes and patterns are the *building blocks* of a VQI. Hence, we review automatic content generation techniques for the *Attribute* and *Pattern Panels*. Although the set of labels of nodes/edges for populating an *Attribute Panel* can be easily generated by traversing the underlying graph repository, automatically generating the patterns of a *Pattern Panel* is an NP-hard problem [24]. Hence, in the sequel, we focus on this. We begin by classifying the patterns into two categories, *basic* and *canned*. Next, we summarize the desirable characteristics of canned patterns [24, 48]. Then, we group the review of data-driven selection of canned patterns into two categories. Finally, we summarize the usability evaluation reported in the literature that highlights the superiority of data-driven VQIs to manual VQIs primarily due to the exposition of diverse canned patterns by the former.

Basic and canned patterns. Intuitively, we can classify the patterns in a *Pattern Panel* into two types, *basic* and *canned*. A *basic* (a.k.a *default*) pattern is a small-size pattern with size at most z (typically, $z \leq 3$) such as edge, 2-edge, and triangle. End users are typically aware of these generic topologies as they are either building blocks of any graph-structured data or they are well-known in a specific domain. On the other hand, a *canned* pattern is a subgraph of size larger than z. These larger size patterns are highly desirable as they often reveal structures that are unique to the underlying graph data source, thereby furnishing representative objects to end users to trigger efficient visual graph query formulation even when they may not have a specific pattern-in-their-head. Hence, in the sequel we focus on canned patterns.

Cognitive psychology-aware canned patterns. Typically, a VQI has limited space for displaying patterns. Furthermore, a long list of patterns may increase browsing time of end users for pattern selection during visual query formulation. Indeed, several industrial VQIs [3, 4, 6] only display a handful of patterns. In particular, recent research [24, 25, 48] proposed that any canned pattern set for a VQI should satisfy the following characteristics in order to assist end users in visual query formulation.

High coverage. A canned pattern p *covers* a graph G if G contains a subgraph s that is isomorphic to p. Since p may have many embeddings in G, the canned pattern set in a *Pattern Panel* should ideally cover as large portion of G as possible. Then a large number of subgraph queries on the underlying graph repository can be constructed by utilizing the pattern set.

High diversity. High coverage of canned patterns is insufficient to facilitate efficient visual query formulations [24]. In order to make efficient use of the limited display space in a VQI, the patterns in a *Pattern Panel* should be *structurally diverse* to serve a variety of queries. This also facilitates bottom-up search where a user gets a bird's-eye view of the diverse substructures in the underlying graph repository. For instance, the canned patterns in [3] have low diversity and hence not a superior enabler of bottom-up search.

Low cognitive load. In cognitive psychology, *cognitive load* refers to the used amount of working memory resources (*i.e.,* memory demand) [44]. In particular, *intrinsic* cognitive load is the effort associated with a specific topic (*e.g.,* selecting patterns) and *extraneous* cognitive load refers to the way information or tasks are presented to a learner (*e.g.,* presentation of patterns in a VQI) [44]. In the context of visual query formulation, cognitive load on a user is associated with browsing a pattern set and visually interpreting a displayed pattern's edge relationships to determine if it is useful for a query. In particular, large graphs overload the human perception and cognitive systems, resulting in poor performance of tasks such as identifying edge relationships [23, 50]. Consequently, a topologically complex pattern may demand substantial cognitive effort (*i.e.,* increase intrinsic cognitive load) from an end user to decide if it can assist in her query formulation [24]. While basic patterns impose low cognitive load due to their small size, it is desirable for canned patterns to be of low cognitive load as well to facilitate cognitively-efficient browsing and selection of relevant patterns during query formulation.

We review measures proposed in the literature [24, 48] to quantify coverage, diversity, and cognitive load of canned patterns and summarize their properties.

Canned pattern selection for a set of small- or medium-sized data graphs. Recent work [24, 45] have proposed novel frameworks for selecting canned patterns for a graph database containing a large collection of small- or medium-sized graphs (*e.g.,* chemical compounds, protein structures). We review them in this section.

The Catapult framework [24] comprises of the following three steps. First, it partitions a collection of data graphs into a set of clusters. Then, it summarizes each cluster into a *cluster summary graph* (CSG) by performing *graph closure* iteratively on pairs of data graphs in the cluster. A *closure graph* [22] *integrates* graphs of varying sizes into a single graph by inserting dummy vertices or edges with a special label such that every vertex and edge is represented in it. Finally, it follows a greedy iterative approach based on *weighted random walks* for selecting the canned patterns from CSGs based on the aforementioned characteristics. Specifically, it exploits a *pattern score* that incorporates coverage, diversity, and cognitive load to associate a score to each candidate pattern. The candidate pattern with the largest pattern score and is within a user-specified size range (*i.e.,* budget) is greedily selected as the best pattern to be added to the canned pattern set of a *Pattern Panel*. The selection process continues until either the required number (a user-defined value in the budget) of canned patterns are discovered or when no new pattern can be found.

Tzanikos *et al.* [45] propose a novel modular architecture to address this problem by dividing it into independent tasks that can be optimized and adapted as needed. For a given graph database, first the similarity score between the graphs is computed. Then this score is used to partition the graphs into clusters, which are then

merged into one *continuous graph*. Finally, the continuous graph is used to extract the canned patterns. The architectural innovation here is that each of these modules can utilize customized solutions and current state-of-art techniques for superior performance.

These frameworks are query log-oblivious primarily due to the lack of publicly-available log data for graph databases.

Canned pattern selection for large networks. The aforementioned techniques for selecting canned patterns from a collection of small- or medium-sized data graphs cannot be utilized for large networks as the clustering-based approach is prohibitively expensive [48]. Here we review a framework called TATTOO [48] to address this problem for large networks. Given a large network G and a user-specified budget b on the number of canned patterns to display and their minimum and maximum permissible sizes, the goal is to automatically select canned patterns for the *Pattern Panel* from G that satisfy b. It exploits a recent analysis of real-world query logs [14] to *classify* topologies of canned patterns into *categories* that are consistent with the topologies of real-world queries (*e.g.,* star, chain, petals, flower). Such classification enables it to bypass the stumbling block of the lack of availability of query logs but yet exploit topological characteristics of real-world queries to guide the selection process. Since real-world query logs contain triangle-like and non-triangle-like substructures, it first decomposes the input network into a dense *truss-infested region* (G_T) and a sparse *truss-oblivious region* (G_O) by leveraging the notion of k-truss [46]. Then candidate patterns from G_T and G_O are discovered based on the classified topologies to identify potentially useful patterns. Lastly, canned patterns are selected from these candidates for display on a VQI based on a novel *pattern set score* that is sensitive to coverage, diversity, and cognitive load of patterns. Specifically, the selection algorithm guarantees $\frac{1}{e}$-approximation.

Usability results. Lastly, we summarize usability evaluations of data-driven VQIs w.r.t its manual counterparts. We discuss it from two dimensions, *performance measures* and *preference measures*. The former types are quantifiable measures (*i.e.,* can be communicated with numbers) whereas the latter ones give an indication of a "user's opinion about the interface which is not directly observable" [32] (through questionnaires and interviews). Data-driven VQIs are reported [24, 48] to be more efficient (lesser query formulation time and number of steps) compared to several industrial-strength manual VQIs. It is also reported to provide a superior experience (preference measures). Hence, they typically outperform manual VQIs in several usability criteria such as efficiency, errors, satisfaction, and flexibility (Section 2.1). Note that these usability evaluations in existing literature are conducted on a small number of end users.

2.4 Data-driven Maintenance of VQIs

Given that the maintenance of node/edge labels in an *Attribute Panel* is straightforward, recent work have focused on maintaining the canned patterns in a *Pattern Panel* as the underlying graph repository D evolves in order to continuously support efficient visual query formulation. In this part, we review a framework to address the *canned pattern maintenance* problem [25] for a large collection of small- or medium-sized data graphs.

The straightforward approach of selecting canned patterns repeatedly using aforementioned frameworks as D evolves to maintain the pattern set can be extremely inefficient [25]. MIDAS [25] addresses this limitation with an effective and efficient canned pattern maintenance technique that is built on top of CATAPULT [24]. Specifically, it seeks to update the existing canned patterns in a VQI such that the updated set continues to have high coverage, high diversity, and low cognitive load. In particular, MIDAS guarantees that the quality of the updated pattern set is at least the same or better than the original canned patterns.

MIDAS [25] maintains the canned patterns based on batch updates instead of unit updates. This is because (a) unit update involves a single data graph and is unlikely to impact the set of canned patterns in a VQI and (b) several real-world databases of small- or medium-sized data graphs are updated periodically (*e.g.,* daily). In particular, it exploits the degree of changes to *graphlet frequency* distribution in D to *selectively* maintain the pattern set. It also replaces frequent subtrees with *frequent closed trees* (FCT) [13] as feature vectors for clustering in CATAPULT. As FCT displays closure property, it paves the way for efficient maintenance of the clusters.

First, MIDAS assigns all newly added graphs to existing clusters of D and removes all graphs marked for deletion. Then, it calculates graphlet frequency distributions for D and the updated version of D. Next, it performs FCT maintenance by first retrieving the existing FCTs and changes to D and then maintaining them due to these changes. The modified clusters and CSGs are maintained after that. MIDAS computes the Euclidean distance between the graphlet distributions of D and updated D to determine the *type* of modification and corresponding action. For major modification, it generates candidates patterns from CSGs of newly-generated and modified clusters. The existing canned patterns are then updated using a *multi-scan swapping strategy* that guarantees progressive gain of coverage without sacrificing diversity and cognitive load. To this end, it leverages on a *coverage-based pruning* strategy and two indices to facilitate pruning of unpromising candidate patterns for selecting new canned patterns. In the case of minor modification, no pattern maintenance is required. Only the underlying clusters and CSGs are maintained to ensure that they are consistent with the updated D.

Usability results. We review the usability study reported in [25]. Specifically, MIDAS can reduce the number of formulation steps and query formulation time compared to manual VQIs, thereby positively impacting several aforementioned usability criteria.

2.5 Future Directions

While good progress has already been made, the research on data-driven visual query interfaces has just begun, and there are many opportunities for continued research. The final part of the tutorial presents open problems (non-exhaustive list) in this area. Some of these topics were introduced by a vision paper [10]. Our grand vision is a pervasive desire to continue stimulating our shift in our traditional thinking by shifting the generation of visual query interfaces from manual to data-driven mode.

Data-driven VQI maintenance for large networks. The research on maintenance of canned pattern set with the evolution of underlying graph repository is still in its nascent stage. Efficient maintenance of VQIs for large networks is still an open problem. Note that a solution to this needs a rethink as the evolution characteristics of large networks differ fundamentally from a collection of data graphs. In the latter case, the repositories are typically updated periodically whereas large networks often evolve continuously.

Data-driven VQIs for massive networks. All research related to data-driven visual query interface construction and maintenance have focused either on a large set of small or medium-sized data graphs or on networks with millions of nodes. Both these types of data are assumed to reside in a single commodity machine. A natural extension to this paradigm is to support similar problems on massive graphs which demands a distributed framework and novel construction and maintenance algorithms built on top of it.

Towards aesthetics-aware data-driven VQIs. An issue that is paramount to an end user but widely ignored by the data management community is the aesthetics of a VQI layout. In fact, as mentioned in Section 2.1, it is one of the usability criteria. Many HCI studies have asserted a strong link between visual complexity and aesthetics [9, 34] and have attempted to measure aesthetics automatically [31, 35, 40]. The work in [33, 35] proposed an array of *aesthetic metrics* to quantify visual complexity such as visual clutter, color variability, contour congestion, and layout quality. Note that visual complexity impacts cognitive load on end users.

The key components that influence visual complexity in a VQI are the *Attribute*, *Pattern*, and *Results Panels*. Manual VQIs work out all the aesthetic issues associated with these panels manually resulting in designs that may not always be aesthetically pleasing (*e.g.,* [6]). Although cognitive load has been considered for canned pattern selection and maintenance in existing work on data-driven VQIs, it is only exploited at selecting individual patterns. The cognitive load imposed by the layout choices of canned patterns and node/edge attribute labels in a VQI has not been explored yet. Furthermore, aesthetically-pleasing and cognitive load-aware presentation of query results in a *Results Panel* is largely unexplored. If a result subgraph containing matches to a user query looks like a hairball in a *Results Panel* then it is hard for an end user to explore it and gain insights from it. In summary, the layouts of existing visual query interfaces are not automatically generated by considering various *aesthetic metrics* and their impact on cognitive load of end users. *Hence, how can we extend data-driven VQI construction techniques to be 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/cognitive load (measured using aesthetics metrics) of the interface.

Beyond Graphs. While this tutorial focuses on data-driven VQIs for graphs, it is easy to see that this paradigm is potentially relevant for other data types where visual querying is prevalent. For example, there are several efforts toward sketch-based querying of time series (*i.e.,* data series) data [17, 29, 36]. Finding patterns of interest in a large collection of such time series data during query formulation can be time-consuming. Hence, a data-driven sketch-based query interface construction framework may potentially mitigate this challenge.

Beyond VQIs. Lastly, the canned pattern selection and maintenance algorithms reviewed in this tutorial have potential use cases beyond data-driven VQIs. For example, given that these patterns have high coverage and diversity, and low cognitive load, they can be potentially useful for efficiently generating graph summaries that are visualization-friendly [28]. Due to cognitive load-conciousness of these patterns in contrast to topological summaries generated by classical graph summarization techniques, they are potentially more palatable to end users.

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 graph data management and HCI in [11] primarily focuses on visual graph query processing and results exploration. That is, it *assumes* a VQI (manual or data-driven) is available for subgraph query formulation. Our tutorial is orthogonal to this effort as we focus on a comprehensive review of the data-driven paradigm of VQI construction and maintenance.

4 BIOGRAPHIES

Sourav S. Bhowmick is an Associate Professor at the School of Computer Science and Engineering (SCSE), Nanyang Technological University, Singapore. His core research expertise is in data management, human-data interaction, and data analytics. He is a co-recipient of Best Paper Awards in ACM CIKM 2004, ACM BCB 2011, and VLDB 2021 for work on mining structural evolution of tree-structured data, generating functional summaries, and scalable attributed network embedding, respectively. Sourav is serving as a member of the SIGMOD Executive Committee, a regular member of the PVLDB advisory board, and a co-lead in the committee for Diversity and Inclusion in Database Conference Venues. He is a co-recipient of the VLDB Service Award in 2018 from the VLDB Endowment. He was inducted into Distinguished Members of the ACM in 2020.

Byron Choi is the Associate Head and an Associate Professor at the Department of Computer Science, Hong Kong Baptist University (HKBU). His research interests include graph-structured databases, database usability, database security, and time series analysis. Byron's publications have appeared in premium venues such as TKDE, VLDBJ, SIGMOD, PVLDB/VLDB, and ICDE. He was awarded a distinguished program committee (PC) member from ACM SIGMOD 2021 and a best reviewer award from ACM CIKM 2021. He received the distinguished reviewer award from PVLDB 2019. 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.

Acknowledgements. Sourav S Bhowmick is supported by the AcRF Tier-2 Grant MOE2015-T2-1-040. Byron Choi is supported by HKRGC GRF 12201119 and 12201518, and IRCMS/19-20/H01. We would also like to acknowledge Huey-Eng Chua (NTU), Kai Huang (NTU & Fudan University), Zifeng Yuan (NTU & Fudan University), and Zekun Ye (NTU & Fudan University) for their contributions to the research of data-driven VQIs.

REFERENCES

[1] The Aesthetic-usability effect. Nielsen Norman Group. https://www.nngroup.com/articles/aesthetic-usability-effect/.
[2] Bloom VQI. https://neo4j.com/bloom.
[3] Drugbank VQI. https://www.drugbank.ca/structures/search/small_molecule_drugs/structure.
[4] eMolecules VQI. https://www.emolecules.com/.
[5] Graph Database Market. *MarketsandMarkets*. https://www.marketsandmarkets.com/Market-Reports/graph-database-market-126230231.html?gclid=Cj0KCQiAxc6PBhCEARIsAH8Hff1pUb5PI2peZmHQa-AvoPd2MRWXyPwGfEKYFu6I86Z-SgGyQ2a8G88aAmgmEALw_wcB, *Last accessed 15th March, 2022.*
[6] PubChem VQI. https://pubchem.ncbi.nlm.nih.gov/edit3/index.html.
[7] S. S. Bhowmick, B. Choi. Plug-and-Play Visual Query Interfaces for Graphs. *To Appear in Synthesis Lectures on Data Management*, Morgan & Claypool Publishers, 2022.
[8] D. Benyon, P. Turner. Designing Interactive Systems: A Comprehensive Guide to HCI and Interaction Design. 2nd edn. *Pearson Education Ltd.*, Edinburgh, 2005.
[9] D. Berlyne. Studies in the new Experimental Aesthetics. Washington D.C., *Hemisphere Pub. Corp.*, 1974.
[10] S. S. Bhowmick, B. Choi, C. E. Dyreson. Data-driven Visual Graph Query Interface Construction and Maintenance: Challenges and Opportunities. *PVLDB* 9(12), 2016.
[11] S. S. Bhowmick, B. Choi, C. Li. Graph Querying Meets HCI: State of the Art and Future Directions. In *SIGMOD*, 2017.
[12] Sourav S. Bhowmick, Kai Huang, Huey Eng Chua, Zifeng Yuan, Byron Choi and Shuigeng Zhou. AURORA: Data-driven construction of visual graph query interfaces for graph databases. In *ACM SIGMOD*, 2020.
[13] A. Bifet, R. Gavaldà. Mining Frequent Closed Trees in Evolving Data Streams. *Intell. Data Anal.*, 15(1):29-48, 2011.
[14] A. Bonifati, W. Martens, T. Timm. An Analytical Study of Large SPARQL Query Logs. *PVLDB*, 11(2), 2017.
[15] A. Bonifati, G. H. L. Fletcher, H. Voigt, N. Yakovets. Querying Graphs. *Synthesis Lectures on Data Management*, Morgan & Claypool Publishers, 2018.
[16] D.H. Chau, C. Faloutsos, H. Tong, et al. GRAPHITE: A Visual Query System for Large Graphs. *In ICDM Workshop*, 2008.
[17] M. Correl, M. Gleicher. The Semantics of Sketch: Flexibility in Visual Query Systems for Time Series Data. In *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2016.
[18] A. De Angeli, A. Sutcliffe, J. Hartmann. Interaction, Usability and Aesthetics: What Influences Users' Preferences? *In Proc. of Conference on Designing Interactive Systems*, 2006.
[19] A. Dix, J. Finlay, G. Abowd, R. Beale. Human-computer Interaction, 2nd edn, *Pearson Education Ltd*, Harlow, 1998.
[20] D. Erdös, Z. Fekete, A. Lukács. Visualized Subgraph Search. *IEEE VAST*, 2009.
[21] F. Haag, S. Lohmann, S. Bold, T. Ertl. Visual SPARQL Querying based on Extended Filter/flow Graphs. *In AVI*, 2014.
[22] H. He, A.K. Singh. Closure-tree: An Index Structure for Graph Queries. *In ICDE*, 2006.
[23] W. Huang, P. Eades, S.-H. Hong. Measuring Effectiveness of Graph Visualizations: A Cognitive Load Perspective. *Information Visualization* 8(3), 2009.
[24] K. Huang, H.-E. Chua, S. S. Bhowmick, B. Choi, S. Zhou. CATAPULT: Data-driven Selection of Canned Patterns for Efficient Visual Graph Query Formulation. In *SIGMOD*, 2019.
[25] K. Huang, H.-E. Chua, S. S. Bhowmick, B. Choi, S. Zhou. MIDAS: Towards Efficient and Effective Maintenance of Canned Patterns in Visual Graph Query Interfaces. In *SIGMOD*, 2021.
[26] K. Huang, S. S. Bhowmick, S. Zhou, B. Choi. PICASSO: Exploratory Search of Connected Subgraph Substructures in Graph Databases. *PVLDB*, 10(12), 2017.
[27] N. Jayaram, S. Goyal, C. Li. VIIQ: Auto-Suggestion Enabled Visual Interface for Interactive Graph Query Formulation. *In PVLDB*, 8(12), 2015.
[28] A. Khan, Sourav S. Bhowmick, F. Bonchi. Summarizing Static and Dynamic Big Graphs. PVLDB 10(12): 1981-1984, 2017.
[29] D.J.L. Lee, J. Lee, T. Siddiqui, J. Kim, K. Karahalios, A.G. Parameswaran. You Can't Always Sketch What you Want: Understanding Sensemaking in Visual Query Systems. *IEEE Trans. Vis. Comput. Graph.*, 26(1): 1267-1277, 2020.
[30] S. Liu, J. P. Cedeno, K. Selçuk Candan, M. L. Sapino, S. Huang, X. Li. R2DB: A System for Querying and Visualizing Weighted RDF Graphs. *In ICDE*, 2012.
[31] P. Mbenza, N. Burny. Computing Aesthetics of Concrete User Interfaces. *In EICS*, 2020.
[32] D. D. McCracken, R. J. Wolfe. User-Centered Website Development: A Human-computer Interaction Approach. *Pearson Education Inc.*, New Jersey, 2004.
[33] A. Miniukovich, A. De Angeli. Quantification of Interface Visual Complexity. *In Working Conference on Advanced Visual Interfaces*, 2014.
[34] A. Miniukovich, M. Marchese. Relationship Between Visual Complexity and Aesthetics of Webpages. *In CHI*, 2020.
[35] A. Miniukovich, A. De Angeli. Computation of Interface Aesthetics. *In SIGCHI*, 2015.
[36] M. Mannino, A. Abouzied. Expressive Time Series Querying with Hand-drawn Scale-free Sketches. *In CHI*, 2018.
[37] A. Oliva, M. L. Mack, M. Shrestha, A. Peeper. Identifying the Perceptual Dimensions of Visual Complexity of Scenes. *In Proc. of the 26th Annual Meeting of the Cognitive Sc. Society*, 2004.
[38] R. Pienta, A. Tamersoy, A. Endert, et al. VISAGE: Interactive Visual Graph Querying. *In AVI*, 2016.
[39] R. Pienta, F. Hohman, et al. Visual Graph Query Construction and Refinement. *In SIGMOD*, 2017.
[40] K. Reinecke, T. Yeh, et al. Predicting Users' First Impressions of Website Aesthetics with a Quantification of Perceived Visual Complexity and Colorfulness. *In SIGCHI*, 2013.
[41] S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, M. T. Özsu. The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing. *PVLDB*, 11(4), 2017.
[42] B. Shneiderman, C. Plaisant. Designing the User Interface: Strategies for Effective Human-Computer Interaction. 5th Ed., Addison-Wesley, 2010.
[43] S. Sun, Q. Luo. In-memory Subgraph Matching: An In-depth Study. In *SIGMOD*, 2020.
[44] J. Sweller, J. van Merrienboer, F. Paas. Cognitive Architecture and Instructional Design. *Educational Psychology Review.* 10 (3): 251–296, 1998.
[45] M. Tzanikos, M. Krommyda, V. Kantere. A Highly Modular Architecture for Canned Pattern Selection Problem. *In DEXA*, 2021.
[46] J. Wang, J. Cheng. Truss Decomposition in Massive Networks. *PVLDB* 5(9), 2012.
[47] S. Yang, Y. Xie, Y. Wu, T. Wu, H. Sun, J. Wu, X. Yan. SLQ: A User-friendly Graph Querying System. *In SIGMOD*, 2014.
[48] Z. Yuan, H.-E. Chua, S. S. Bhowmick, Z. Ye, W.-S. Han, B. Choi. Towards Plug-and-play Visual Graph Query Interfaces: Data-driven Canned Pattern Selection for Large Networks. *PVLDB*, 14(11), 2021.
[49] Z. Yuan, H.-E. Chua, S. S. Bhowmick, Z. Ye, B. Choi, W.-S. Han. PLAYPEN: Plug-and-play Visual Graph Query Interfaces for Top-down and Bottom-up Search on Large Networks. *In SIGMOD*, 2022.
[50] V. Yoghourdjian, D. Archambault, S. Diehl, T. Dwyer, K. Klein, H. C. Purchase, and H.-Y Wu. Exploring the Limits of Complexity: A Survey of Empirical Studies on Graph Visualization. *Visual Informatics* 2(4), 2018.
[51] 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.
[52] L. Zou, L. Chen, J.X. Yu, Y. Lu. A Novel Spectral Coding in a Large Graph Database. *In EDBT*, 181-192, 2008.