FGreat: Focused Graph Query Autocompletion

Nathan Ng*, Peipei Yi†, Zhiwei Zhang*, Byron Choi*, Sourav S. Bhowmick‡ and Jianliang Xu*
*Department of Computer Science, Hong Kong Baptist University, Hong Kong
†Lenovo Machine Intelligence Center, Hong Kong
‡ School of Computer Science and Engineering, Nanyang Technological University, Singapore
*{cskhng,cszwzhang,bchoi,xujl}@comp.hkbu.edu.hk, †pyi@lenovo.com ‡assourav@ntu.edu.sg

***Abstract*—Composing queries is evidently a tedious task. This is particularly true of graph queries as they are typically complex and prone to errors. This is compounded by the fact that graph schemas can be missing or too loose to be helpful for query formulation. *Graph Query AutoCompletion* (GQAC) alleviates users from the potentially painstaking task of graph query formulation. This demonstration presents an interactive visual *Focused* GRaph quEry AutocompleTion framework, called FGREAT. Its novelty relies on the *user focus* for GQAC, which is a subgraph of the current query that a user is focusing on. FGREAT automatically computes a focus and completes the query at the focus, as opposed to an arbitrary query subgraph. This demonstration presents two complementary approaches to compute the user focus for different circumstances. It computes the focus from either (i) the sequence of edges that a user recently added to his/her query, or (ii) the position of the mouse cursor, if it is available. We demonstrate that the user focus enhances *both the effectiveness and efficiency* of graph query autocompletion.**

***Index Terms*—Subgraph Query, graph autocompletion, graphs, database usability**

I. Introduction

Recently, there has been an explosive growth of graph-structured data in a variety of domains such as bioinformatics, collaboration networks, and co-purchase networks. Consequently, it is paramount to develop user-friendly and efficient tools to construct queries for the underlying graph data. Being a proficient query writer is essential for formulating graph queries. Learning a graph query language, however, can be a difficult task. Hence, visual tools are proposed to provide users with a handy way to interactively formulate the queries. Nonetheless, query formulation under a visual environment still requires significant human efforts. As a result, *graph query autocompletion framework* (GQAC) is proposed to alleviate the burdens of graph query formulation. For illustration, this demonstration considers subgraph queries (*i.e.*, the query formalism is *subgraph isomorphism*), but other queries that are of graph structures (such as subgraph homomorphism and simulation queries) can be seamlessly supported.

GQAC takes the user's current query (a.k.a. initial query) as input and adds subgraph increments to the query graph to form *query suggestions*. The query suggestions are ranked according to the user preferences. The user may adopt one of the returned query suggestions and iteratively invoke GQAC to complete their queries. Despite the initial progress of GQAC, according to the research on human-computer interaction (HCI), humans can only focus on several recent software artifacts in hand [1]. That is, users may only work on certain portions of the query graph. It implies that many suggestions generated by extending the query at its arbitrary subgraphs could be irrelevant. Returning such suggestions may adversely affect the effectiveness of query autocompletion and even distract users. We illustrate this in Example 1.1.

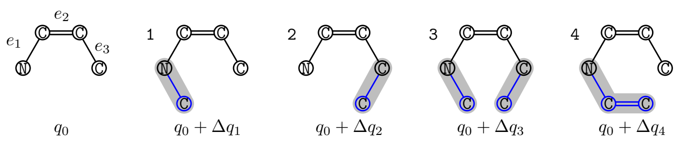


Fig. 1. An example of suggestions from GQAC

Example 1.1: Consider a publicly available chemical database (*e.g.*, PUBCHEM,[1] AIDS[2], and EMOL[3]). Suppose a user wishes to search for compounds containing the Pentadienylamine[4] substructure. The partial subgraph query constructed by him/her is q_0 in Figure 1. The top-4 suggestions returned from GQAC are $q_0 + \Delta q_1$ to $q_0 + \Delta q_4$. Observe that each suggestion is composed by adding small increments (highlighted in blue with gray background) to the query graph. He/she may select a useful suggestion (if present) by clicking on it, thus saving mouse clicks to manually formulate the new nodes and edges. Suppose the user formulates q_0 in the order of e_1, e_2, and e_3. The user may focus on the most recent edges (*i.e.*, e_2 and e_3). Only $q_0 + \Delta q_2$ (extended from e_3) is a relevant suggestion as it is a subgraph of pentadienylamine. The user can simply click on the suggestion to adopt it instead of drawing Δq_2 manually.

This paper demonstrates, for the first time, a novel and interactive *Focused* GRaph quEry AutocompleTion framework, called FGREAT. A user focus is a subgraph of the query that has the highest user attention, and FGREAT attempts to automatically complete the query at the user focus. FGREAT realizes the concept of user focus by two approaches, namely GFOCUS [6] and MFOCUS, which take different information as input and could be applied in different circumstances. GFOCUS requires user's *implicit* information, *i.e.*, it does not assume any specific input devices (*e.g.*, touch screens), while MFOCUS assumes users move the mouse cursor to their focus.

- The GFOCUS approach is the first work that leverages on the temporal and structural locality principles inspired by HCI research to derive the user focus. Intuitively, when users

[1] `https://pubchem.ncbi.nlm.nih.gov/search/`
[2] `https://dtp.cancer.gov/databases_tools/bulk_data.htm`
[3] `https://www.emolecules.com/`
[4] `https://pubchem.ncbi.nlm.nih.gov/compound/59750537`

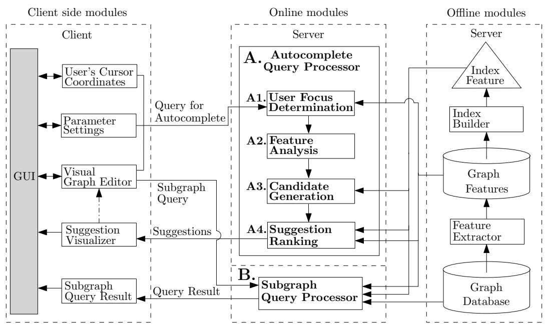


Fig. 2. Architecture of FGREAT

formulate queries by adding edges and nodes, or adopting query suggestions, their attention localizes in those few recent edges/subgraphs. GFOCUS takes user's query formulation sequence (*i.e.*, the user actions in constructing the query) as input and computes the user focus according to such locality principles. The user focus is *automatically* computed and maintained *in the background*.

• The second approach MFOCUS determines the user focus using the mouse cursor position in the visual environment. It takes the position of both query graph and mouse cursor as input, calculates the distances between the cursor and each query node, then predicts the possible user focus. By using this approach, FGREAT can easily detect arbitrary shifts of user focus or even no focus.

After FGREAT obtained the user focus using either GFOCUS or MFOCUS, it computes candidate query suggestions (by adding subgraph increments to the focus). FGREAT presents a new greedy suggestion ranking algorithm that is aware of both user focus and user preferences [6]. The top-k suggestions are returned to users. This demonstration is the first to illustrate that the user focus can enhance not only the effectiveness but also the efficiency of GQAC.

Related work. There have been three recent research studies on GQAC [3]–[5]. Pienta et al. [4] and Li et al. [3] have demonstrated interactive methods to produce *edge* or *node suggestions*. In contrast, only FGREAT and AUTOG [5] offer *subgraph suggestions*. However, only FGREAT exploits user focus. Some GUIs of publicly available databases (*e.g.*, PUBCHEM and EMOL) have hard-coded subgraph query templates. However, they lack GQAC functionalities.

II. SYSTEM ARCHITECTURE

The FGREAT demonstration adopts a simple client-server architecture. Figure 2 shows the major modules of the architecture. The modules at the client-side are online, whereas the modules at the server-side can be further divided into the online and offline ones.

1. The user interface (UI) on the client-side. The client-side modules mainly provide a user interface of FGREAT. Figure 3 depicts the screenshot of the UI of FGREAT. Panel 1 provides various target graph databases for users to choose. It lists a set of node labels and edge labels for users to construct their graph queries. The *Visual Graph Editor* (Panel 2) is a canvas for constructing query graphs. Users may click on an

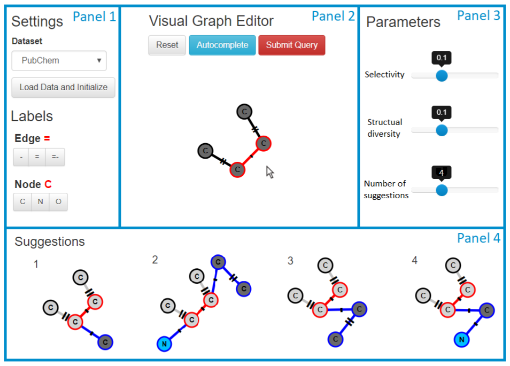


Fig. 3. The UI of FGREAT, where the user focus is highlighted in red in the Visual Graph Editor (Panel 2)

empty space of the editor to add a new node or drag from one node to another to form a new edge. During the query graph formulation process, one may obtain query suggestions by clicking the *Autocomplete* button to invoke GFOCUS, or moving the cursor to the part that he/she wishes to extend and invoke MFOCUS.[5] Information such as the coordinates of the mouse cursor and the query graph are sent to the server-side for query autocompletion. In Panel 3, users may use the slides to indicate their preferences on query suggestions. After the suggestions are ranked, the *Suggestion Visualizer* (Panel 4) displays the top-k suggestions in real time. To show different components of the suggestions, the visualizer highlights the user focus in *red* and the subgraph increments in *blue*. The existing query is colored in light grey. If a suggestion is useful, users may adopt it by simply clicking on it. When the *Submit Query* button in Panel 2 is clicked, the query is evaluated. The *Subgraph Query Results* are retrieved from the server, then displayed in the visual interface.

2. Online modules on the server-side. The *Autocomplete Query Processor* on the server-side takes the current query, user's information (*e.g.*, user's past actions and the cursor position) from the client-side as input and produces query suggestions as output. It includes the following four modules.

A1. User focus determination. This module determines the user focus of the query graph.

The MFOCUS approach: The main idea is that users may manually move the mouse cursor to the part that he/she wants to extend. The server simply calculates the distance between the cursor and each query node and sets the node (denoted as v_f) with the smallest distance as a part of the user focus.

The GFOCUS approach: Users construct their queries step by step. GFOCUS uses a set of operators that aim at capturing generic, fundamental actions during the construction for determining the focus. The set of fundamental query formulation operators of this demonstration is OP = {add, adopt, rollback,

[5]In practice, the suggestions can be regularly and automatically generated without any user input. However, for the ease of user study, FGREAT requires users to click the *Autocomplete* button for suggestions or press *ENTER* to capture the mouse location. Hence, users are aware of the query autocompletion step to provide their qualitative evaluations on FGREAT.

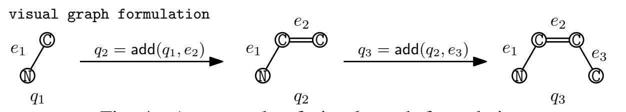


Fig. 4. An example of visual graph formulation

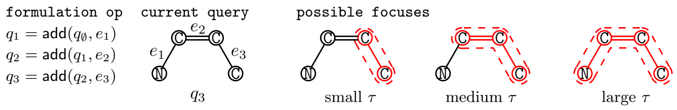


Fig. 5. Formulation of q_3 and possible focuses by varying τ

click}. For better readability, we skip the definitions of these operators.

A query formulation sequence is a sequence of query formulation operators. A query formulation sequence S_t at the t-th step is denoted as follows:

$$q_t = \mathsf{op}_t(\ldots \mathsf{op}_k(\ldots \mathsf{op}_1(q_0))), \text{where } \mathsf{op} \in \mathsf{OP}. \quad (1)$$

To model the users' attention during query formulations, GFocus proposes to assign an *attention weight* to each query edge. When an edge is added to q, it has a weight 1. GFocus brings two major results from HCI research to the weight maintenance:

1) *Temporal locality.* Berman et al. [2] observed that memory fades due to the mere passage of time. In query formulation, GFocus adopts the exponential decay function as the forgetting function of the weights. Users' attention localizes on the small number of newly operated edges.
2) *Structural locality.* When humans are working on a software artifact, their attention is naturally localized on, and then propagated to, a small number of neighboring artifacts [1]. Hence, when formulating queries, the edges that are close to the newly operated edges receive higher users' attention and are more likely to be a part of the user focus. In GFocus, each neighboring edge of the last operated edges receives a portion of weight that is positively proportional to their structural proximity.

We propose a parameter τ to model user's memory strength. Figure 5 shows the formulation of q_3. The larger the value of τ, the larger the focus. Intuitively, the user focus is the subgraph having the maximal attention density that the user remembers (see [6] for definitions).

Example 2.1: Suppose a user is composing a query graph q_3 manually from q_1, shown in Figure 4. He/she may obtain the query q_3 via the query formulation sequence below:

$$q_3 = \mathsf{add}(\mathsf{add}(e_1, \text{“-”}, e_2), \text{“=”}, e_3),$$

where e_1, e_2, and e_3 are edges in the query. The labels of the nodes are marked in the circles (*i.e.*, “C”), and the labels of the edges are signified by “=” and “-” connecting the circles. The user's focuses of different τ values are highlighted in red.

A2. Feature analysis. Features are generally known to be subgraphs that carry important characteristics of the graph database. Without prior information, we assume queries may carry such characteristics. FGreat indexes features that are connected to form large subgraphs and hence, larger queries that are formed from smaller ones, for efficient online query autocompletion [5]. Some feature examples of a real-world

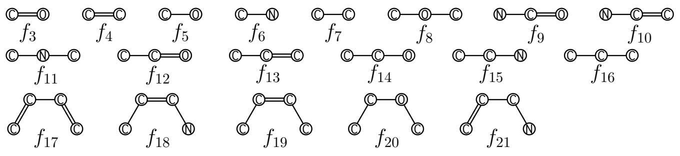


Fig. 6. Some features (*e.g.*, frequent subgraphs) of a publicly available chemical database

graph (chemical) database is shown in Figure 6. This module decomposes the user's query into a set of features and their *embeddings* (*i.e.*, the locations) in the query. Once the query is represented as such, GFocus uses the subgraph s_f with the maximal normalized density of user attention weights as the user focus. On the other hand, MFocus selects the smallest feature that contains v_f as the user focus.

A3. Suggestion candidate generation. The input of the *Candidate Generation* module is the decomposed query q and a user focus. This module outputs a set of candidate query suggestions. Intuitively, a *candidate query suggestion* is formed by adding a subgraph increment (which is a feature mined offline) to the user focus of q. Different from [5], query increments are only added to the focus, but not arbitrary places of q. We adopt a technique of our previous work [5] to efficiently prune automorphic suggestions and suggestions that will not retrieve any data graphs.

A4. Suggestion ranking. The number of candidate query suggestions is exponential to the query graph size. Furthermore, the size of the suggestion panel is limited, and users may only interpret a small set of the query suggestions. Hence, FGreat returns only top-k suggestions. Without any prior knowledge, such as user's behavioral profile, we propose the following generic ranking function util. It should be remarked that util is only for illustration purposes (*i.e.*, other functions can be readily plugged into the FGreat framework).

Given a suggestion set Q': $\{q'_1, q'_2, \ldots, q'_k\}$ and user preference α and β, the *user intent value* of Q' (util) is defined as follows:

$$\mathsf{util}(Q') = \frac{\alpha}{k} \sum_{q' \in Q'} \mathsf{sel}(q') + \frac{\beta}{k} \sum_{q' \in Q'} \delta_{\mathsf{inc}}(q') + (1 - \alpha - \beta)\mathsf{coverage}(Q'),$$

where $\alpha \in [0, 1], \beta \in [0, 1]$ and $(1 - \alpha - \beta) \in [0, 1]$.

• The sel function is the total result count of $q' \in Q'$. α specifies how much a user prefers to obtain query suggestions that have high result counts.

• The $\delta_{\mathsf{inc}}(q')$ function is defined as $\frac{1}{|\Delta(q')|}$. β refers to the preference on the suggestions constructed from small subgraph increments. It is noted from experiments that small suggestions can be easier to predict than those from large subgraph increments, while large correct suggestions save more manual formulation effort.

• The diversity of a suggestion set Q' is determined by coverage. A union graph of all possible suggestions can be considered as the universe of all possible suggestions [6]. The coverage function $\mathsf{coverage}(Q')$ counts the number of edges of the universe that are covered by Q'.

Example 2.2: Figure 7 shows a current query q and candidate suggestions q'_1, q'_2, and q'_3. Assume the dashed area is the user focus. The right-hand side of Figure 7 is the union

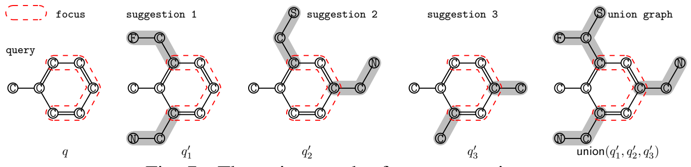


Fig. 7. The union graph of two suggestions

graph of q'_1, q'_2, and q'_3. In addition to the edges of the focus, $\{q'_1, q'_2\}$ covers 7 edges of the union and $\{q'_1, q'_3\}$ covers only 5 edges. Hence, $\{q'_1, q'_2\}$ has a higher coverage value.

III. Experimental Results

We have performed a user study and an experiment to investigate the performance of FGreat. Our results reflect that identifying user focus can improve not only the effectiveness but also the efficiency of the previous system AutoG [5].

Suggestion quality via user study. We conducted a user study on FGreat to survey whether the computed user focus was accurate and if FGreat helped the users or not. We mimicked the GUI (Figure 3) and followed the settings of AutoG. We conducted the test with 21 volunteers with computer science background. Each volunteer was given size 8-edge target queries and was asked to formulate them with FGreat, which highlights the computed user focus in red, and the volunteers were not aware of the details. After formulating each query, they reported their levels of agreement to the following two statements:

1) "*The red colored edges capture the portion that I am working on.*"; and
2) "*The suggestions are useful when I draw my query.*"

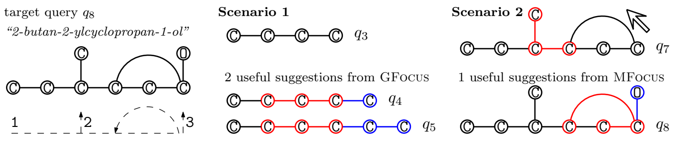
The average user satisfaction of the determined user focus was consistently high (4.4 of 5). Consistent with [5], we verified that the *total profit metric* (TPM) value is an indicator of suggestions qualities (the correlation coefficient with the user's satisfaction score is 0.93 and the p-value is 0.007). For queries of hi, mid and low TPM values of AutoG, FGreat obtained 3.85, 3.30 and 2.10 user satisfactions, whereas AutoG obtained 4.55, 2.95 and 1.65 only. AutoG has 4.55 for hi TPM queries because they are their best queries [5].

We have also invited 3 chemists to use FGreat. They generally agree with the first statement above. Moreover, TPM still indicates the suggestion qualities.

Simulated query formulation steps. We investigated the qualities of the suggestions of FGreat through large-scale simulations under a variety of settings. In a nutshell, we simulate the process of a user using FGreat to construct a graph query from scratch. We adopted several popular metrics for evaluating suggestion qualities [3], [5]. The experimental results show that the suggestions of FGreat often have higher TPMs than those of AutoG. Furthermore, FGreat is 35 times more efficient than AutoG on average. For further details, please refer to the technical report [6].

IV. Demonstration overview

The key objective of the demonstration is to let the attendees interactively experience the *focused* graph query autocompletion. In our demonstration, one will be able to formulate his/her queries manually or by adopting query suggestions. We describe two scenarios with a chemical compound "*2-Butan-2-ylcyclopropan-1-ol*", *i.e.*, the target query q_8 (Figure 8). The graph with dashed lines under the target query illustrates the formulation order that the user plans. It is natural that a user first draws the long chain (marked as 1), and then the short branches (marked as 2 and 3).


target query q_8
"2-butan-2-ylcyclopropan-1-ol"
1
2
3
Scenario 1
q_3
2 useful suggestions from GFocus
q_4
q_5
Scenario 2
q_7
1 useful suggestions from MFocus
q_8
Fig. 8. The chemical query q_8 used in the demo scenarios

Scenario 1. Query autocompletion at user focus without any explicit user feedback. When the current query is q_3, GFocus returns two useful suggestions (*i.e.*, q_4 and q_5, suggested increments are highlighted in blue). The user focus computed by GFocus using the query formulation sequence is highlighted in red. All suggestions enlarge q_3 at the user focus. The attendees can get a feel for the usefulness of user focus. In contrast, no suggestions returned by AutoG are useful as suggestions are generated at arbitrary places of q_3. The demonstration shows faster response times of GFocus because fewer candidates are generated at the focus.

Scenario 2. Query autocompletion at mouse cursor. We present this scenario with a current query q_7. The user focus determined by GFocus is highlighted in red in q_7. The user focus is inaccurate since the user did a "random" focus shift to draw branch 3 after formulating branch 2, which is predicted by GFocus. Hence, GFocus does not return useful suggestions. In this case, the user can switch to MFocus. The focus is highlighted in red in q_8. One may verify that MFocus correctly determined the user focus and it returns a useful suggestion (*i.e.*, the target query q_8).

Finally, the attendees can observe the suggestion characteristics by tuning system parameters (*e.g.*, α, β and k). A demonstration video is publicly available at https://goo.gl/gX6LSn.

Acknowledgments. This work was partially supported by AcRF MOE2015-T2-1-040, NSFC 61602395, and HKRGC GRF12232716, 12201315, 12201518, and 12258116. Mr. Ng visited NTU in Jun-July 2017 under the UG overseas summer research scholarship of Dept. of CS at HKBU.

References

[1] A. Baddeley, M. Eysenck, and M. Anderson. *Memory*. Cognitive Psychologie. Psychology Press, 2009.
[2] M. G. Berman, J. Jonides, and R. L. Lewis. In search of decay in verbal short-term memory. *J Exp Psychol Learn Mem Cogn*, page 317, 2009.
[3] N. Jayaram, S. Goyal, and C. Li. VIIQ: Auto-suggestion enabled visual interface for interactive graph query formulation. *PVLDB*, pages 1940–1951, 2015.
[4] R. Pienta, F. Hohman, A. Tamersoy, A. Endert, S. B. Navathe, H. Tong, and D. H. Chau. Visual graph query construction and refinement. In *SIGMOD*, pages 1587–1590, 2017.
[5] P. Yi, B. Choi, S. S. Bhowmick, and J. Xu. Autog: a visual query autocompletion framework for graph databases. *VLDB J.*, 26(3):347–372, 2017.
[6] P. Yi, B. Choi, Z. Zhang, S. S. Bhowmick, and J. Xu. Gfocus: User focus-based graph query autocompletion. https://goo.gl/MYYw94, 2018.